



Triakis Corporation

**Ancillary Simulator
Parts Document**

For the

**Shuttle Remote
Manipulator System**

**A NASA CI03
SARP Initiative 583
IVV-70 Project**



Table of Contents

1	Introduction	4
1.1	System purpose	4
1.2	System scope	4
1.3	Definitions, acronyms, and abbreviations.....	4
1.4	References	5
1.5	SRMS overview	5
2	General system description	6
3	System performance characteristics.....	7
3.1	System context.....	7
3.2	Major system components	8
3.3	Major system capabilities	8
3.4	System hardware design	9
3.5	Ancillary parts	9
3.5.1	RMS Control Panel.....	10
3.5.2	Remote Manipulator Arm	18
3.5.3	Shuttle Bay.....	22
3.5.4	Part Descriptions	23

Table of Figures

Figure 1:	Simulated RMA Within Shuttle Orbiter	7
Figure 2:	Simulator RMS Control & Display Panel.....	8
Figure 3:	Shuttle RMS Block Diagram	10



Table of Tables

Table 1:	RMS Control Panel Part Hierarchy	10
Table 2:	RMS Control Panel SPI Bus Output Data.....	11
Table 3:	Code Example for reading RMS Control Panel Output Data.....	12
Table 4:	RMS Control Panel SPI Bus Input Data	14
Table 5:	ES Code Example for Sending Data to RMS Control Panel Part	15
Table 6:	RMS Control Panel Part Summary.....	15
Table 7:	ES Code Example for Sending Video Data to RMS Control Panel	16
Table 8:	Remote Manipulator Arm Part Hierarchy	18
Table 9:	Shuttle Bay Part Hierarchy	22
Table 10:	AFDX Messages from RMS Computer to Data Modules	29
Table 11:	AFDX Messages from Data Module to the RMS Computer.....	29
Table 12:	Code Example for Sending Messages to a Data Module	29
Table 13:	Code Example for Receiving Messages From a Data Module	30
Table 14:	AFDX Messages from RMS Computer to an Image Sensor	40
Table 15:	AFDX Messages from an Image Sensor to the RMS Computer.....	40
Table 16:	Code Example for Sending Messages to an Image Sensor.....	40
Table 17:	ES Code Example for Receiving Messages From an Image Sensor	41
Table 18:	AFDX Messages from RMS Computer to a Camera Lamp	42
Table 19:	ES Code Example for Sending Messages to a Camera Lamp	42
Table 20:	AFDX Messages from RMS Computer to a Motor Controller.....	46
Table 21:	AFDX Messages from Motor Controller to the RMS Computer	46
Table 22:	Code Example for Sending Messages to a Motor Controller	47
Table 23:	Code Example for Receiving Messages From a Motor Controller	47



1 Introduction

This specification is being developed to support a research project funded by the NASA Software Assurance Research Program (SARP) during the fiscal year 2003 Center Initiatives (CI03) effort. A system-level, executable specification (ES) based simulation of the Shuttle Remote Manipulator System (SRMS) has been created from the requirements specified in the System Requirements (SARP-I583-001) and Simulator Requirements (SARP-I583-002) Specifications, and will be used as a vehicle for exploring the concepts described in section 2 of Triakis proposal number TC_G020614.

This document describes the ancillary parts developed to simulate the system design described in the System Design Document (SARP-I583-101). The parts described herein are those peripheral to the RMS computer, and therefore common to both the ES and DE simulator environments.

The simulator created for this project will be used to evaluate the extent to which the Triakis concept of Executable Specifications (ES') achieves unambiguous communication of system requirements thereby reducing errors induced by interpretation of ambiguous specifications. It will also be used to evaluate the potential that substituting a DE in place of the ES, has for reducing costs and maintaining test consistency through reuse of unmodified system level tests.

Further, new methods of gathering software metrics through use of the simulator will be sought, explored, and evaluated. The virtual system simulator developed for this project will be used to evaluate other potential benefits that its virtual system integration laboratory (VSIL) environment offers in support of general testability, independent validation & verification (IV&V), reliability, and safety.

1.1 System purpose

The system specified herein is intended to represent the SRMS in a general sense only. The ancillary described in this document will be an integral element of the virtual system simulator that will be used as a vehicle to facilitate the research goals stated in Triakis proposal number TC_G020614. System components and functions of the real-world SRMS that are not required to support our research goals have been omitted.

While the purpose of the actual SRMS is to facilitate the deployment and retrieval of shuttle payloads as well as extra-vehicular activity missions, the derivative SRMS will not incorporate functioning end-effectors required for these purposes. The specified SRMS will demonstrate limited control and movement capability of the RMA along with simulated cameras and video monitors showing the RMA position.

1.2 System scope

The SRMS approximately models a subset of the system characteristics of the existing NASA space shuttle RMS. Adaptations to the functionality of the actual SRMS have been incorporated to the extent required for the stated research purposes and demonstration of the research results.

1.3 Definitions, acronyms, and abbreviations

AFDX	Avionics Full Duplex Switched Ethernet
CCTV	Closed-Circuit Television
CI03	Center Initiative for fiscal year 2003
C/W	Caution/Warning
DE	Detailed Executable



ES	Executable Specification
IV&V	Independent Verification and Validation
N/A	Not Applicable
NASA	National Aeronautics & Space Administration
OSMA	Office of Safety and Mission Assurance
PDRS	Payload Deployment and Retrieval System
RHC	Rotational Hand Controller
RMA	Remote Manipulator Arm
RMS	Remote Manipulator System
RMSC	RMS Computer
RMSCP	RMS Control Panel
SARP	Software Assurance Research Program
SimRS	Simulator Requirements Specification
SRMS	Shuttle Remote Manipulator System
SyDD	System Design Document
SyRS	System Requirements Specification
THC	Translational Hand Controller
VSIL	Virtual System Integration Laboratory

1.4 References

- <http://science.ksc.nasa.gov/shuttle/technology/sts-newsref/sts-deploy> NASA PDRS web page
- SARP-I583-001 System Requirements Specification for the Shuttle Remote Manipulator System
- SARP-I583-002 Simulator Requirements Specification for the Shuttle Remote Manipulator System
- SARP-I583-101 System Design Document for the Shuttle Remote Manipulator System
- SARP-I583-201 Hardware Design Document for the Shuttle Remote Manipulator System
- TC_G020614 Triakis proposal to NASA for the SARP (Solicitation No: NRA SARP 0201), 14 June 2002

1.5 SRMS overview

Please refer to the NASA [PDRS](#) web page for a more complete description of the real space shuttle SRMS that this system is designed to resemble. The following excerpt is included for quick reference:

The payload deployment and retrieval system (PDRS) includes the electromechanical arm that maneuvers a payload from the payload bay of the space shuttle orbiter to its deployment position and then releases it. It can also grapple a free-flying payload, maneuver it to the payload bay of the orbiter and berth it in the orbiter. This arm is referred to as the remote manipulator system (RMS).

The shuttle RMS is installed in the payload bay of the orbiter for those missions requiring it. Some payloads carried aboard the orbiter for deployment do not require the RMS.

The RMS is capable of deploying or retrieving payloads weighing up to 65,000 pounds. The RMS can also be used to retrieve, repair and deploy satellites; to provide a mobile extension ladder for extravehicular activity crew members for work stations or foot restraints; and to be used as an inspection aid to allow the flight crew members to view the orbiter's or payload's surfaces through a television camera on the RMS.

While this specification addresses requirements for the simulator itself, those requirements are, to a great extent driven by the system requirements. Consequently, the following excerpt from the NASA [PDRS](#) web page has been included here to provide an overview of the real-world [SRMS](#):



2 General system description

The simulator designer used the following excerpt from the [NASA PDRS web page](#) as a reference source and it is given here to provide a general [SRMS](#) description for informational purposes only.

The basic [RMS](#) configuration consists of a manipulator arm; an [RMS](#) display and control panel, including rotational and translational hand controllers at the orbiter aft flight deck flight crew station; and a manipulator controller interface unit that interfaces with the orbiter computer. Normally, only one [RMS](#) is installed during a shuttle mission, on the left longeron of the orbiter payload bay.

The [RMS](#) arm is 50 feet 3 inches long, 15 inches in diameter, and has six degrees of freedom. The six joints of the [RMS](#) correspond roughly to the joints of the human arm with shoulder yaw and pitch joints; an elbow pitch joint; and wrist pitch, yaw and roll joints. The end effector is the unit at the end of the wrist that actually grabs, or grapples, the payload.

The [RMS](#) can only be operated in a zero gravity environment, since the arm dc motors are unable to move the arm's weight under the influence of Earth's gravity. Each of the six joints has an extensive range of motion, allowing the arm to reach across the payload bay, over the crew compartment or to areas on the undersurface of the orbiter. Arm joint travel limits are annunciated to the flight crew arm operator before the actual mechanical hard stop for a joint is reached.

One flight-crew member operates the [RMS](#) from the aft flight deck control station, and a second flight-crew member usually assists with television camera operations. This allows the [RMS](#) operator to view [RMS](#) operations through the aft flight deck payload and overhead windows and through the closed-circuit television monitors at the aft flight deck station.

The orbiter's [CCTV](#) aids the flight crew in monitoring [PDRS](#) operations. The arm has provisions on the wrist joint for a [CCTV](#) camera that can be zoomed, a viewing light on the wrist joint and a [CCTV](#) with pan and tilt capability on the elbow of the arm. In addition, four [CCTV](#) cameras in the payload bay can be panned, tilted and zoomed. Keel cameras may be provided, depending on the mission payload. The two [CCTV](#) monitors at the aft flight deck station can each display any two of the [CCTV](#) camera views simultaneously with split screen capability. This shows two views on the same monitor, which allows crew members to work with four different views at once. Crewmembers can also view payload operations through the aft flight station overhead and aft (payload) viewing windows.

The arm has a number of operating modes. Some of these modes are computer-assisted, moving the joints simultaneously as required to put the end point (the point of resolution, such as the tip of the end effector) in the desired location. Other modes move one joint at a time; e.g., single mode uses software assistance and direct and backup hard-wired command paths that bypass the computers.

Four [RMS](#) manually augmented modes are used to grapple a payload and maneuver it into or out of the orbiter payload retention fittings. The four manually augmented modes require the [RMS](#) operator to use the [RMS](#) translational hand controller (THC) and rotational hand controller (RHC) with the computer to augment operations.

The [THC](#) and [RHC](#) located at the aft flight deck station are used exclusively for [RMS](#) operations. The [THC](#) is located between the two aft viewing windows. The [RHC](#) is located on the left side of the aft flight station below the [CCTV](#) monitors. The [THC](#) and [RHC](#) have only one output channel per axis. Both [RMS](#) hand controllers are proportional, which means that the command supplied is linearly proportional to the deflection of the controller.

There are two types of automatic modes that can be used to position the [RMS](#) arm: operator-commanded and preprogrammed.

The operator-commanded automatic mode moves the end effector from its present position and orientation to a new one defined by the operator via the keyboard and [RMS](#) CRT display. The arm moves in a straight line to the desired position and orientation and then enters the hold mode.



The preprogrammed auto sequences operate in a manner similar to the operator-commanded sequences. Instead of the [RMS](#) operator entering the data on the computer via the keyboard and CRT display, the [RMS](#) arm is maneuvered according to a command set programmed before the flight, called sequences. Each sequence is an ordered set of points to which the arm will move. Up to 200 points may be preprogrammed into as many as 20 sequences.

The description provided is intended to give a general picture of system functionality upon which our virtual system has been modeled. The features actually implemented and the fidelity of this virtual [SRMS](#) representation have been chosen according to what is needed to support our overall research goals.

Unless otherwise indicated, subsequent references to all elements of the [SRMS](#) and surrounding systems within this document are to be construed as referring to the virtual system elements within the simulator being developed and not the actual [SRMS](#) (in use on the NASA shuttle program) on which the virtual system is based.

3 System performance characteristics

3.1 System context

The [SRMS](#) described in the [SyRS](#) is designed as a self-contained system with few connections to the virtual shuttle within which it will function. [Figure 1](#) shows a screenshot of the simulated [RMA](#) within the virtual shuttle orbiter. Neither the manipulator positioning mechanism nor a functioning end effector will be implemented in this [SRMS](#).

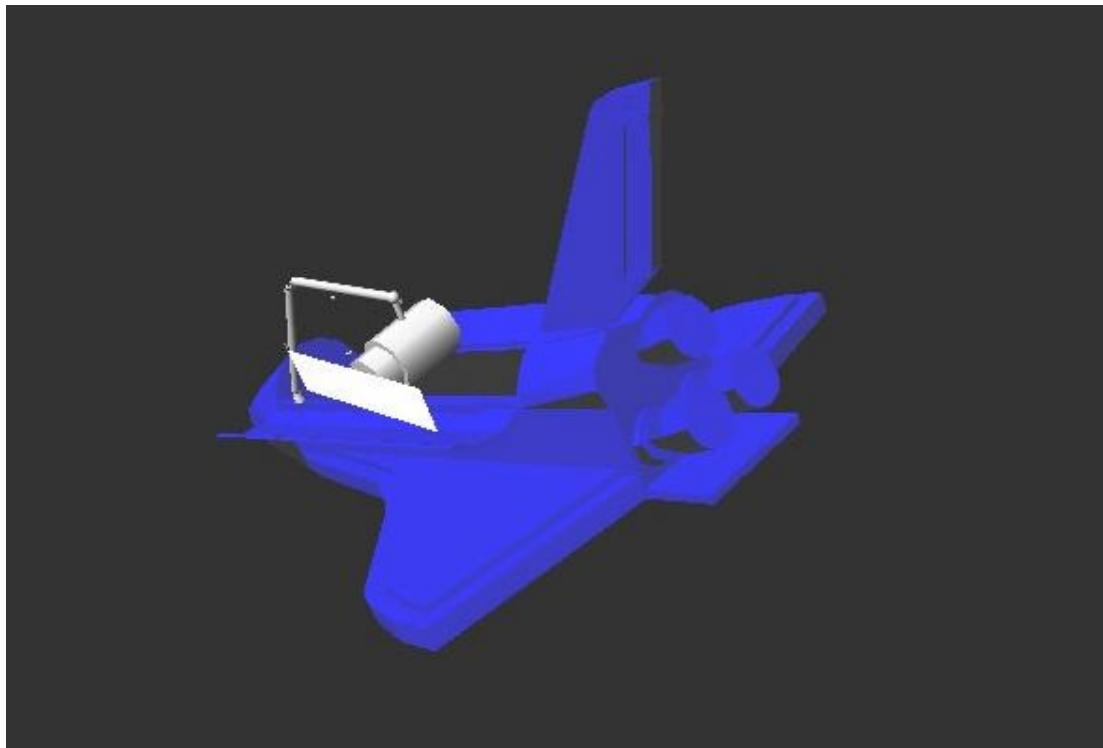


Figure 1: Simulated RMA Within Shuttle Orbiter

The [RMA](#) is attached to the portside cargo door support longeron in the shuttle orbiter cargo bay as depicted in [Figure 1](#).



The SRMS draws its power from the space shuttle 28VDC and 115VAC/400Hz power supplies as required to function as described herein.

The RMS control & display panel and the closed circuit television (CCTV) monitors that the crew employs in the operation of the SRMS are provided as part of the simulator, but not located on a simulated orbiter flight deck at the aft crew station as originally specified in the SyRS. Instead, the RMS control & display panel will be rendered in a simulator window through which the operator may operate and monitor the SRMS.

3.2 Major system components

The SRMS comprises three principal elements:

- a) A remote manipulator arm (RMA) ([Figure 1](#)),
- b) A RMS control & display panel ([Figure 2](#)), and
- c) A RMS control computer (RMSCC).

CCTV monitors have been simulated for visually monitoring RMA activity during operation.

The RMSCC provides the interface between the RMS control & display panel and the RMA itself.

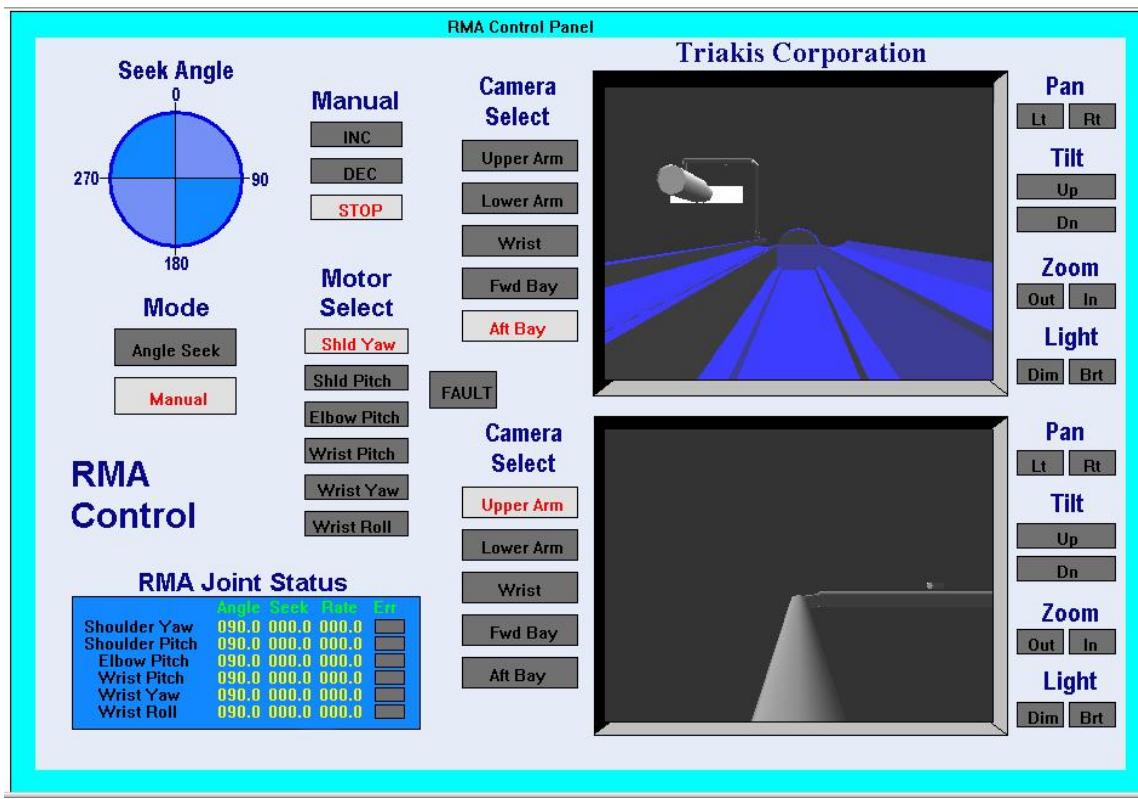


Figure 2: Simulator RMS Control & Display Panel

3.3 Major system capabilities

The RMA is implemented with 6 degrees of freedom corresponding roughly to the joints of the human arm i.e.: shoulder yaw & pitch joints; elbow pitch joint; and wrist pitch, yaw, & roll joints.



Both the upper and lower RMA booms are equipped with strain gauge sensors to measure the dynamic forces exerted on them during operation.

The SRMS design incorporates five CCTV video cameras as specified in the System Requirements Specification. Each of the cameras is equipped with pan, tilt, and zoom capability in addition to featuring a controllable light source. The cameras are located as stated in the SyRS i.e.:

- One on the RMA upper arm boom,
- One on the RMA lower arm boom,
- One at the RMA wrist joint,
- One at the aft wall of the shuttle bay, and
- One at the forward wall of the shuttle bay.

The RMS Control Panel incorporates two video display monitors and buttons as required for displaying CCTV video from any of the five video cameras.

To the left of each video monitor on the RMS Control Panel are five buttons used to select the desired camera view for display. Camera controls located to the right of the video display monitors on the RMS Control Panel are used for positioning, and zooming the camera whose view has been selected for display. While buttons have been incorporated into the control panel for controlling the lighting level of the selected camera view, the lamp parts have not been programmed to implement that functionality for this project.

3.4 System hardware design

The hardware design for the RMS Computer that has been documented in the SRMS Hardware Design Document (SARP-I583-201) has been developed into a functional DE for use in our research. [Figure 3](#) shows a system-level block diagram of the Shuttle Remote Manipulator System.

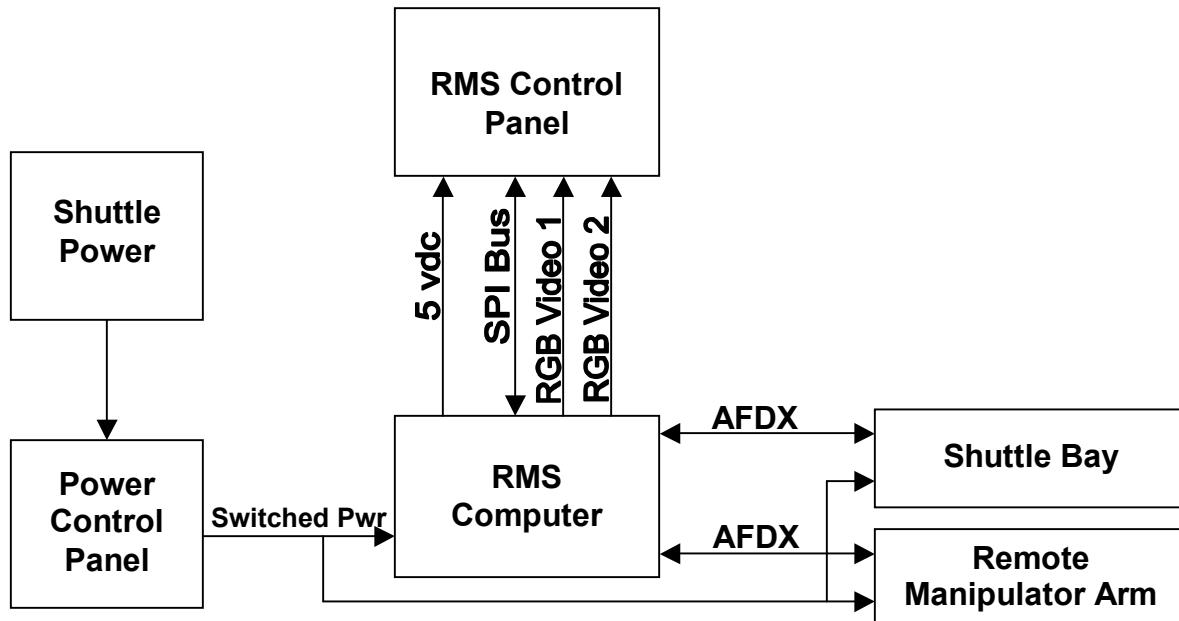
Power is supplied to the SRMS from the 115vac main shuttle avionics power bus via a circuit breaker on the Power Control Panel. The RMS Computer converts the incoming power to DC voltages suitable to power its own electronics as well as those within the RMS Control Panel. Switched power from the power control panel supplies power to the shuttle bay cameras and the Remote Manipulator Arm as well.

The RMS Computer communicates with the Remote Manipulator Arm and the cameras in the shuttle bay via Avionics Full Duplex Switched Ethernet (AFDX) serial high-speed databases. In addition to commands and status information, digital compressed video from the CCTV cameras are conveyed over these databases.

The RMS Computer converts the compressed digital camera video signals into RGB format to drive the video inputs of the two video display monitors located on the RMS Control Panel. The RMS Computer communicates with the RMS Control Panel via a Serial Peripheral Interface bus.

3.5 Ancillary parts

The ancillary parts described in the following sections are those with significant functionality. Many lesser ancillary parts used in the simulator have been created in an attempt to present a more complete system design with a complement of parts that might be found in a real-world implementation of the SRMS design developed for this project. Parts not described include the “Shuttle Power” and “Power Control Panel” (see [Figure 3](#)) for example because our research did not require the development of functionality for these components. Many of the parts used have provisions for redundancy, however, none of the secondary signals and databases have been implemented for this project.

**Figure 3: Shuttle RMS Block Diagram**

The primary subsystems shown in [Figure 3](#) that contain ancillary parts of interest are the “RMS Control Panel,” the “Remote Manipulator Arm,” and the “Shuttle Bay.” Since the only parts of interest (aside from the RMA) in the “Shuttle Bay” are the forward and aft cameras, it will not be discussed independently. The shuttle bay cameras are identical to the three used on the RMA itself.

3.5.1 RMS Control Panel

Because of the relatively large number of parts that typically comprise a simulator, it is usually more convenient to present simulator part hierarchies in table format. Icosim is an object oriented tool and once a part has been created, multiple instances of that part may be defined and applied as required throughout the simulator. [Table 1](#) lists the three parts that comprise the RMS Control Panel. The number in the “Level” column indicates the hierarchy level of the parts listed in the “Instance” column of the table.

Table 1: RMS Control Panel Part Hierarchy

Part Type	Level	Part Instance Name
RMSControlPanel	3	RMS Ctrl Panel
VideoMonitor	4	Video 1
VideoMonitor	4	Video 2
ControlPanelGraph	4	CP Graph

The RMS Control Panel and the Control Panel Graphic parts have been created to operate in a close-coupled configuration. It helps to think of the RMS Control Panel part as equivalent to the interface electronics and the Control Panel Graphic part as equivalent to the panel hardware with the buttons & displays comprising the user interface.

Extending this analogy, the RMS Control Panel part contains all of the interface and control functions for communication with the RMS Computer. It is also responsible for driving control panel indicators & displays, and receiving & processing operator inputs from the control panel. The Control Panel Graphic part then is responsible



for little more than for rendering the panel image, detecting operator inputs, passing them to the RMS Control Panel part, and updating the panel display as commanded by the RMS Control Panel part.

The RMS Control Panel and Control Panel Graphic parts communicate with each other directly through ‘C’ function calls as opposed to standard data busses & signals. This exception from the standard method of creating IcoSim parts was made for expediency since the simulator developed for this project will not be used to build real hardware. A more rigorous system specification for this part might specify the interfaces between the panel interface electronics and the panel user interface hardware (switches, indicators, etc.).

The Video Monitor part class (instantiated as Video 1 & Video 2) converts the camera video signal into a bitmap image that is displayed on the control panel where specified in the Control Panel Graph part. This part class uses the OpenGL graphics libraries for managing the video images.

The following subparagraphs contain additional part details:

3.5.1.1 RMS Control Panel

The RMS Control Panel (depicted in [Figure 2](#)) receives DC power from an external source the RMS Computer and communicates with it via the industry standard serial peripheral interface (SPI) bus. All user activated control inputs are passed to the RMS Computer and displayed data & indicators are driven by inputs received from the RMS Computer via the SPI bus. This implementation of the SPI bus makes use of a dual master scheme where either the RMS Computer or the RMS Control Panel may act as bus master.

[Table 2](#) lists the command and data formats used for communicating over this bus. Packet IDs and data words are 16 bit values. Maximum packet size is 16 words.

Table 2: RMS Control Panel SPI Bus Output Data

Packet ID Data[0]	Data Word No.	Data Definition
0 = Select Mode	Data[1]	0 = Angle Seek command mode 1 = Manual command mode
1 = Select Motor	Data[1]	0 = shoulder yaw motor 1 = shoulder pitch motor 2 = elbow motor 3 = wrist pitch motor 4 = wrist yaw motor 5 = wrist roll motor
2 = Manual Command	Data[1]	0 = Increment 1 = Decrement 2 = Stop 3 = Command Buttons Off
3 = Angle Seek	Data[1]	0 = Shoulder Yaw 1 = Shoulder Pitch 2 = Elbow Pitch 3 = Wrist Pitch 4 = Wrist Yaw 5 = Wrist Roll
	Data[2]	Seek Angle Integer
	Data[3]	Seek Angle Fraction * 4096



Packet ID Data[0]	Data Word No.	Data Definition
4 = Video1 Select	Data[1]	0 = Upper Arm camera 1 = Lower Arm camera 2 = Wrist camera 3 = Forward Bay camera 4 = Aft Bay camera
5 = Video2 Select	Data[1]	0 = Upper Arm camera 1 = Lower Arm camera 2 = Wrist camera 3 = Forward Bay camera 4 = Aft Bay camera
6 = Camera 1 Control	Data[1]	Camera 1 Control Word (packed bits) 1 = Pan Left 2 = Pan Right 4 = Tilt Up 8 = Tilt Down 10 = Zoom In 20 = Zoom Out 40 = Light Dim 80 = Light Bright
7 = Camera 2 Control	Data[1]	Camera 2 Control Word (packed bits) 1 = Pan Left 2 = Pan Right 4 = Tilt Up 8 = Tilt Down 10 = Zoom In 20 = Zoom Out 40 = Light Dim 80 = Light Bright

[Table 3](#) gives an example of the code used to read data from the RMS Control Panel part. Please refer to the full RMS Computer ES code listing for more detail and examples.

Table 3: Code Example for reading RMS Control Panel Output Data

```
void RMSComputer::HandleSignal(int id, Sig *data)
{
    SPISig spisig;
    BoolSig bsig;

    switch(id) {
        // Handle SPI input bus Chip Select Signal

        case _CS_1:
            bsig = *((BoolSig *)data);
            // look for chip select high to low transition to start reading command
            if(_cs_1 && !bsig.tf) {
                wordcount = 0; // Prepare for new Control Panel data packet
            }
    }
}
```



```
_cs_1 = FALSE;
}
// look for chip select low to high transition to execute command
// _CS_1 going high signals end of control panel data transmission
else if(!_cs_1 && bsig.tf) {
    ExecuteCommand(); // Process new input from Control Panel
    _cs_1 = TRUE;
}
break; // End _CS_1

// Handle Incoming Serial SPI data from Control Panel
case SerialChan_1_In:

    // Accumulate maximum of 16 command words
    // (MAX_SPI_WORDS defined in 'RMSComputer.h')
    // all SPI words are 16 bits long
    if(wordcount < MAX_SPI_WORDS)
        spiChan1Data[wordcount++] = (short)((SPISig *)data)->data;

    break; // End SerialChan_1_In
}

void RMSComputer::ExecuteCommand()
{
    unsigned short majorCommandCode;
    short joint;

    majorCommandCode = spiChan1Data[0]; // Retrieve new major command code
    switch(majorCommandCode) {

        case SelectMode: // Process control mode button inputs
            if(mode != spiChan1Data[1]) { // Potential mode change
                modeChange = TRUE;
                mode = spiChan1Data[1];
            }
            break;

        case SelectMotor: // Process joint select button inputs
            selectedMotor = spiChan1Data[1];
            break;

        case ManualCommand: // Process command button inputs
            cmdChange = TRUE;
            manualCmd = spiChan1Data[1];
            break;

        case AngleSeek: // Process angle seek selection inputs
            joint = spiChan1Data[1];
            seekAngle[joint] = spiChan1Data[2];
            seekAngleFract[joint] = spiChan1Data[3] / 4096;
            newSeekAngle[joint] = TRUE;

            break;
    }
}
```



```
// case video1select: etc., etc.  
}  
}
```

[Table 4](#) lists the SPI Bus Packet ID and data sent by the RMS Computer to the RMS Control Panel. The data types shown in the Packet ID column identify the type of data being received. RMS Computer data is used to update the RMA joint status display, control the backlight of the manual control switches, and to light or extinguish the 'Fault' lamp. All data packets consist of five 16-bit words sent in the sequence given in Table 2.

Table 4: RMS Control Panel SPI Bus Input Data

Packet ID Data[0]	Data Word No.	Data Definition
0 = Panel Display Data	Data[1]	Display Row (RMA Joint) 0 = Shoulder Yaw 1 = Shoulder Pitch 2 = Elbow Pitch 3 = Wrist Pitch 4 = Wrist Yaw 5 = Wrist Roll
	Data[2]	Joint Angle Integer
	Data[3]	Joint Angle Fraction * 4096
	Data[4]	Seek Angle Integer
	Data[5]	Seek Angle Fraction * 4096
	Data[6]	Joint Rate Integer
	Data[7]	Joint Rate Fraction * 4096
1 = Manual Command Data	Data[1]	RMA Joint 0 = Shoulder Yaw 1 = Shoulder Pitch 2 = Elbow Pitch 3 = Wrist Pitch 4 = Wrist Yaw 5 = Wrist Roll
	Data[2]	0 = Increment 1 = Decrement 2 = Stop 3 = CmdBtnsOff
2 = Fault Data	Data[1]	Fault Data Word (packed bits: 1 = Fault) 0x 1 = Master Fault 0x 2 = Shoulder Yaw Fault 0x 4 = Shoulder Pitch Fault 0x 8 = Elbow Pitch Fault 0x 10 = Wrist Pitch Fault 0x 20 = Wrist Yaw Fault 0x 40 = Wrist Roll Fault

[Table 5](#) gives an example of the code used to send data to the RMS Control Panel part. Please refer to the RMS Computer ES code (RMSComputer.cpp) for more detail and examples.

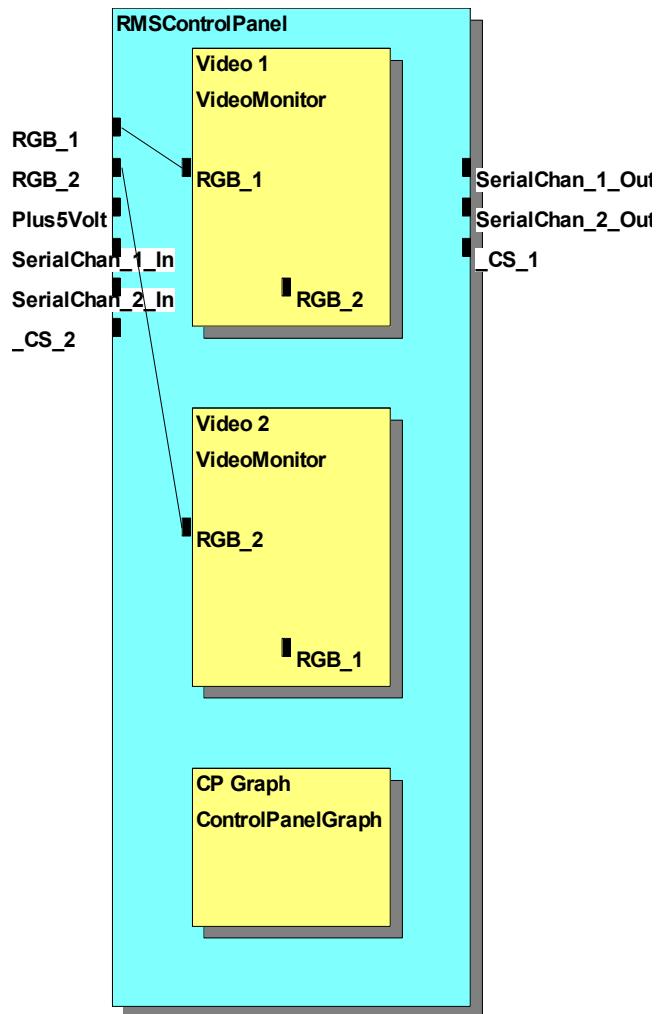
**Table 5: ES Code Example for Sending Data to RMS Control Panel Part**

```
// Tell Control Panel to light 'STOP' command button
bsig.tf = FALSE;
SendSignal(_CS_DATA_1, &bsig);
spisig.data = ManualCmdData;
SendSignal(SerialChan_Data_1_Out, &spisig);
spisig.data = joint;
SendSignal(SerialChan_Data_1_Out, &spisig);
spisig.data = STOP; // Light 'Stop' command button for respective motor
SendSignal(SerialChan_Data_1_Out, &spisig);
bsig.tf = TRUE;
SendSignal(_CS_DATA_1, &bsig);
```

[Table 6](#) summarizes the parts comprising, and the signals used for interfacing with the RMS Control Panel part.

Table 6: RMS Control Panel Part Summary

Class Name	RMSControlPanel	
Sub-Part Class	Sub-Part Instance	
VideoMonitor	Video 1 Video 2	
ControlPanelGraph	CP Graph (includes Switches, Controls, Indicators)	
Signal Input	Signal Output	Signal Type
Plus5Volt		Sig Voltage
SerialChan_Data_1_In	SerialChannel_1_Out	Sig SPI
SerialChan_Data_2_In	SerialChannel_2_Out	Sig SPI
RGB_1		Sig RGB
RGB_2		Sig RGB
_CS_DATA_1	_CS_1	Sig Bool
_CS_DATA_2	_CS_2	Sig Bool
Address/Data Bus	N/A	
Direct Function Call		
Auto Test Function Call		
User Interface Input		
User Interface Monitor		
Internal State Variables		
Requirements	Manual control & display of RMS	
Limitations		



3.5.1.2 Video Monitor

Two Video Monitor parts are used in the RMS Control Panel for display of camera video signals on the control panel graphic part. An example of the ES code used to send the video bitmap data to the Video Monitor parts (passed through the RMS Control Panel part) is given in [Table 7](#). Please refer to the RMS Computer ES code (RMSComputer.cpp) for more detail and examples.

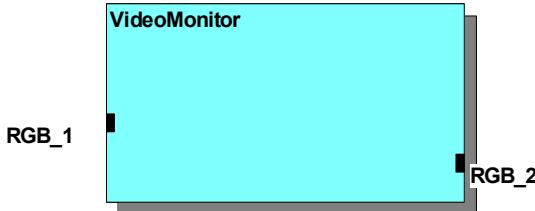
Table 7: ES Code Example for Sending Video Data to RMS Control Panel

```
case Video1Select: // Process new camera source request for display 1
    vidsrc1 = spiChan1Data[1];
    // Point to video buffer for selected camera
    rgbsig.bitmappptr = camBitMap[vidsrc1];
    SendSignal(RGB_1, &rgbsig); // Send video to control panel
    break;
```

Following is a summary of the Video Monitor part signals and sub-parts.

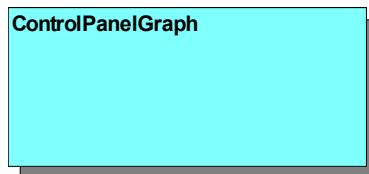


Class Name	VideoMonitor	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
RGB_1		Sig RGB
RGB_2		Sig RGB
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Convert video signal to display graphic	
Limitations	None	



3.5.1.3 Control Panel Graph part summary

Class Name	ControlPanelGraph	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
None	None	
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Enable user input for control & monitoring of the RMS	
Limitations	None	





3.5.2 Remote Manipulator Arm

While the Remote Manipulator Arm contains a great many parts, the number of different part types is comparatively few. [Table 8](#) lists the parts that comprise the Remote Manipulator Arm. The part type names are hyperlinked to the corresponding subparagraph containing a part description (if applicable), summary, and block diagram.

The RMA part itself is a container part passing signals to and from the appropriate subparts. Refer to the RMS Hardware Design Document (SARP-I583-201) for descriptions on how the RMA sub-parts are interconnected. All communication between the RMS Computer and the various RMA sub-parts are conducted via the AFDX serial high-speed databus.

Only the following four RMA sub-part types communicate with the RMS Computer:

- [Motor Controller](#)
- [Image Sensor](#) (camera image)
- [Data Module](#) (strain gauge interface)
- [Lamp](#) (camera lighting)

Information regarding data formats used for communication between the RMS Computer and RMA sub-parts will be covered at the sub-part description level.

Table 8: Remote Manipulator Arm Part Hierarchy

Part Type	Level	Part Instance Name
RemoteManipulatorArm	3	RMA
Shoulder	4	Shoulder
ShoulderPitchMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
ShoulderYawMotor	5	Yaw Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
MotorController	5	Yaw Controller
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Node	5	Node A
Node	5	Node B
UpperArm	4	Upper Arm
Camera	5	Camera
CamMotor	6	Pitch Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear



Part Type	Level	Part Instance Name
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Yaw Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Zoom Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
MotorController	6	Pitch Controller
MotorController	6	Yaw Controller
MotorController	6	Zoom Controller
Lamp	6	Lamp
ImageSensor	6	ImageSensor
SpaceShuttle3DCamera	7	Camera
AFDX_Router	6	Router A
AFDX_Router	6	Router B
Node	6	Node A
Node	6	Node B
AFDX_Router	5	Router_A
AFDX_Router	5	Router_B
StrainGage	5	Strain Gage
DataModule	5	Data Module
Node	5	Node A
Node	5	Node B
Elbow	4	Elbow
ElbowMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Node	5	Node A
Node	5	Node B
LowerArm	4	Lower Arm
Camera	5	Camera
CamMotor	6	Pitch Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope



Part Type	Level	Part Instance Name
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Yaw Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Zoom Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
MotorController	6	Pitch Controller
MotorController	6	Yaw Controller
MotorController	6	Zoom Controller
Lamp	6	Lamp
ImageSensor	6	ImageSensor
SpaceShuttle3DCamera	7	Camera
AFDX_Router	6	Router A
AFDX_Router	6	Router B
Node	6	Node A
Node	6	Node B
AFDX_Router	5	Router_A
AFDX_Router	5	Router_B
StrainGage	5	Strain Gage
DataModule	5	Data Module
Node	5	Node A
Node	5	Node B
Wrist	4	Wrist
WristPitchMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
WristYawMotor	5	Yaw Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
WristRollMotor	5	Roll Motor



Part Type	Level	Part Instance Name
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
MotorController	5	Yaw Controller
MotorController	5	Roll Controller
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Camera	5	Camera
CamMotor	6	Pitch Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Yaw Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Zoom Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
MotorController	6	Pitch Controller
MotorController	6	Yaw Controller
MotorController	6	Zoom Controller
Lamp	6	Lamp
ImageSensor	6	ImageSensor
SpaceShuttle3DCamera	7	Camera
AFDX_Router	6	Router A
AFDX_Router	6	Router B
Node	6	Node A
Node	6	Node B
Node	5	Node A
Node	5	Node B
EndEffector	4	Effector Arm
AFDX_Router	5	Router_A
AFDX_Router	5	Router_B
Ground	5	Ground



3.5.3 Shuttle Bay

[Table 9](#) lists the parts that comprise the Shuttle Bay. The Shuttle Bay part type names are hyperlinked to the corresponding subparagraph containing a part description (if applicable), summary, and block diagram.

As with the RMA, the Shuttle Bay part itself is a container part passing signals to and from the appropriate subparts. Refer to the RMS Hardware Design Document (SARP-I583-201) for descriptions on how the Shuttle Bay sub-parts are interconnected. For the purposes of this document, the only real parts of interest within the Shuttle Bay are the two cameras. All communication between the RMS Computer and the Shuttle Bay camera sub-parts are conducted via the AFDX serial high-speed databus.

Table 9: Shuttle Bay Part Hierarchy

Part Type	Level	Part Instance Name
ShuttleBay	3	Shuttle Bay
Camera	4	BayCamFore
CamMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
CamMotor	5	Yaw Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
CamMotor	5	Zoom Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
MotorController	5	Yaw Controller
MotorController	5	Zoom Controller
Lamp	5	Lamp
ImageSensor	5	ImageSensor
SpaceShuttle3DCamera	6	Camera
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Node	5	Node A
Node	5	Node B
Camera	4	BayCamAft
CamMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic



Part Type	Level	Part Instance Name
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
CamMotor	5	Yaw Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
CamMotor	5	Zoom Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
MotorController	5	Yaw Controller
MotorController	5	Zoom Controller
Lamp	5	Lamp
ImageSensor	5	ImageSensor
SpaceShuttle3DCamera	6	Camera
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Node	5	Node A
Node	5	Node B
AFDX_Router	4	Router A
AFDX_Router	4	Router B

3.5.4 Part Descriptions

The part classes within the part tree hierarchy are described in this section. Each part is identified in a separate subsection with a description comprising the following:

- A high level description of the part
- All sub-parts that make up the part
- A block diagram of the interconnection of the sub-parts
- The functional requirements of the part
- Any special features or fault operation of the part



In the figures supplied for each part, the yellow boxes represent subparts. The top label on the box is the part (instance) name and the next label is the part class. Signals flow into the part are on the left side of the box, and out of the part on the right except for parts at the lowest level where only the pins are identified. Following is a summary list of the unique part classes used in the RMS Control Panel, RMA, and the Shuttle Bay:

AFDX_Router	LowerArm	ShoulderYawMotor
Camera	MotorController	ShuttleBay
CamMotor	MotorGraph	SpaceShuttle3DCamera
ControlPanelGraph	Node	StrainGage
DataModule	OscScope	UpperArm
DCMotor	PlanetaryGearSet	VideoMonitor
Elbow	RemoteManipulatorArm	Wrist
ElbowMotor	RMSControlPanel	WristPitchMotor
EndEffector	ShaftEncoder	WristRollMotor
ImageSensor	Shoulder	WristYawMotor
Lamp	ShoulderPitchMotor	

3.5.4.1 AFDX_Router

Class Name	AFDX Router	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
Databus_1_I	Databus_1_O	Sig AFDX
Databus_2_I	Databus_2_O	Sig AFDX
Databus_3_I	Databus_3_O	Sig AFDX
Databus_4_I	Databus_4_O	Sig AFDX
Databus_5_I	Databus_5_O	Sig AFDX
Databus_6_I	Databus_6_O	Sig AFDX
Power		Sig Bool
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Route AFDX signals to all buses	
Limitations	None	



3.5.4.2 Camera

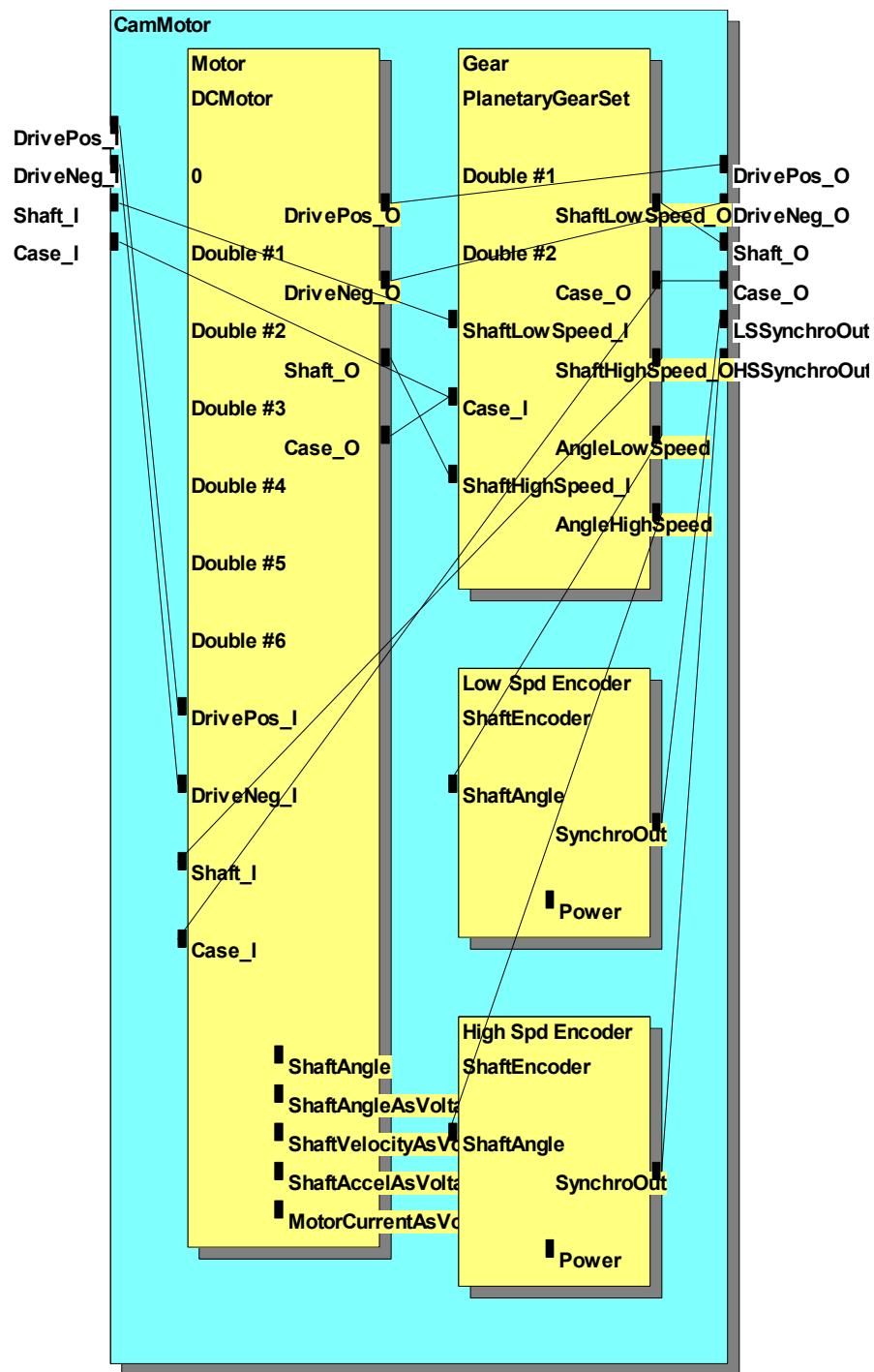
Class Name	Camera	
Sub-Part Class	Sub-Part Instance	
CamMotor	Pitch Motor Yaw Motor Zoom Motor	
MotorController	Pitch Controller Yaw Controller Zoom Controller	
Lamp	Lamp	
Image Sensor	Image Sensor	
AFDX_Router	Router A Router B	
Node	Node A Node B	
Signal Input	Signal Output	Signal Type
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Power_I	Power_O	Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.3 Cam Motor

Class Name	CamMotor	
Sub-Part Class	Sub-Part Instance	
Signal Input	Signal Output	Signal Type
DCMotor	Motor	
PlanetaryGearSet	Gear	
Shaft Encoder	Low Speed Encoder High Speed Encoder	
DrivePos		Sig Thev
DriveNeg		Sig Thev
	Shaft	Sig Force
	Case	Sig Force
	LSSynchroOut	Sig Synchro
	HSSynchroOut	Sig Synchro
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	



3.5.4.4 Data Module

The Data Module is a four-channel analog data acquisition unit used to read the strain gauges and provide the interface to the AFDX bus. [Table 10](#) and Table 12 list the AFDX command and response data formats used for communicating with the Data Module part. As with the SPI bus, command and data words are 16 bit values and the maximum packet size is 16 words.



There are three basic message types used by the RMS Computer to communicate with all AFDX devices:

- a) **Query Status – All:** Requests the status of all AFDX devices;
- b) **Query Response:** Requests data of a specific AFDX device;
- c) **Execute Command:** Issues a command for a specific AFDX device to execute.

AFDX messages from the RMS Computer to the Data Modules take the form shown in [Table 10](#).

Table 10: AFDX Messages from RMS Computer to Data Modules

Message Type	AFDX Dest. Addx	Src Addx Data[0]	Data Type Data[1]	Data Wd Data[2]
1 = Query Status - All	Unused	0x0000	Unused	Unused
2 = Query Response	Device Addx	0x0000	1 = Status	Unused
			7 = Resistance	0 = Ch. No. 0 1 = Ch. No. 1 2 = Ch. No. 2 3 = Ch. No. 3
3 = Execute Command	Device Addx	0x0000	2 = Self Test	Unused

The Data Module sends messages only in response to commands from a controlling device (e.g. RMS Computer). The format of the Data Module response messages is shown in [Table 11](#). The Message Type field contains the code for the RMS Computer message for which the response has been generated.

Table 11: AFDX Messages from Data Module to the RMS Computer

Message Type	AFDX Dest. Addx	Src. Addx Data[0]	Data Type Data[1]	Status Wd Data[2]	Data Wd Data[3]	Data Wd Data[4]
1 = Query Status - All	0x0000	Device Addx	1 = Status	16-bit status word	Unused	Unused
2 = Query Response	0x0000	Device Addx	1 = Status	16-bit status word	Unused	Unused
			7 = Resistance	16-bit status word	0 = Ch. No. 0 1 = Ch. No. 1 2 = Ch. No. 2 3 = Ch. No. 3	Resistance (16-bit Integer)
			3 = Cmd Error	16-bit status word	Ch. No.	1 = Invalid Ch. No.
3 = Execute Command	0x0000	Device Addx	1 = Status	16-bit status word	Unused	Unused

[Table 12](#) gives an example of the code used to send AFDX messages to the Data Module part. Please refer to the full RMS Computer ES code listing for more detail and examples.

Table 12: Code Example for Sending Messages to a Data Module

```
// Send AFDX commands to query strain gauge data modules
for(ga = 0; ga < NumStrainGauges; ga++) {
    afdxsigout.msgType = AFDXSig::QueryResponse; // Command type
    afdxsigout.address = strainGaugeAFDXAddr[ga]; // Set AFDX Addx
    afdxsigout.data[1] = Resistance; // Data type
    afdxsigout.data[2] = 0; // Analog channel (Ch 0 only one currently used)
    SendSignal(Databus_A_O, &afdxsigout); // Issue command
}
```



[Table 13](#) gives an example of the code used to read data from the Data Module part. Please refer to the RMS Computer ES code (RMSComputer.cpp) for more detail and examples.

Table 13: Code Example for Receiving Messages From a Data Module

```

case Databus_A_I: // Handle all incoming AFDX data
    afdxsigin = *((AFDXSig *)data);

    // Check for data from strain gauge modules
    for(ga = 0; ga < NumStrainGauges; ga++)
        if(afdxsigin.data[0] == strainGaugeAFDXAddr[ga]) { // Address match?
            switch(afdxsigin.data[1]) { // What type of data is it?

                case Status: // Status data
                    strnGaModuleStatus[ga] = afdxsigin.data[2];
                    break; //End Status

                case Resistance: // Resistance data
                    strnGaModuleStatus[ga] = afdxsigin.data[2];
                    analogChanNum = afdxsigin.data[3];
                    strnGaData[ga][analogChanNum] = afdxsigin.data[4];
                    break; //End Resistance

                case CmdError: // Command error
                    strnGaModuleStatus[ga] = afdxsigin.data[2];
                    analogChanNum = afdxsigin.data[3]; // Analog channel number
                    errorMsg = afdxsigin.data[4]; // Error code
                    break; // End CmdError
            }
        }
    }
}

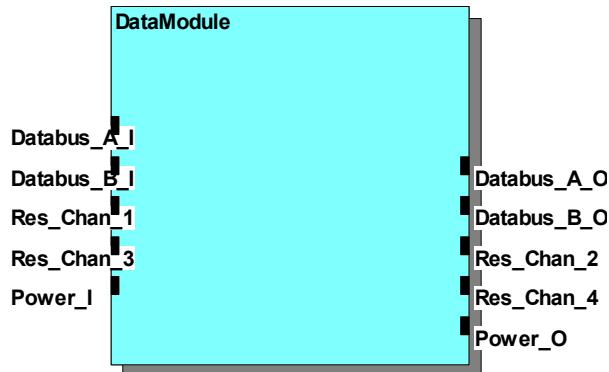
```

The Data Module part summary table and part block diagram are given below:

Class Name	DataModule	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Res_Chan_1		Sig Voltage
Res_Chan_2		Sig Voltage
Res_Chan_3		Sig Voltage
Res_Chan_4		Sig Voltage
Power		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	AFDXSig afdxout	



	double chanVoltage[4]
Requirements	<ol style="list-style-type: none"> 1. Handle AFDX Commands QueryStatusAll QueryResponse, return Strain Gage resistance 2. Receive Strain Gage signal 3. Handle Power On/Off
Limitations	None

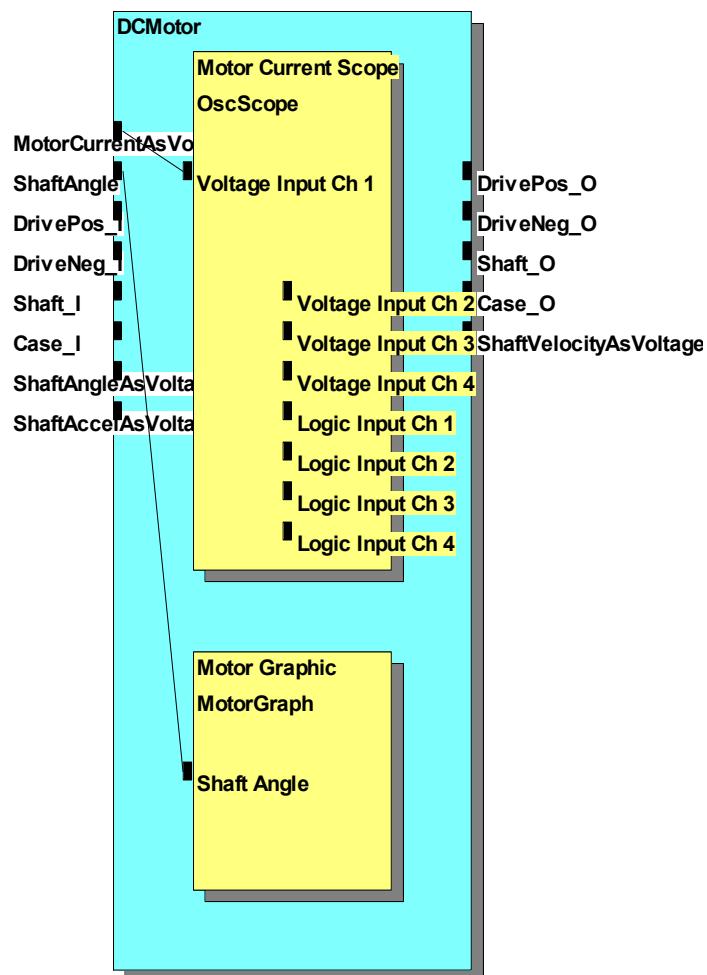


3.5.4.5 DC Motor

Class Name	DCMotor	
Sub-Part Class	Sub-Part Instance	
OscScope	Motor Current Scope	
MotorGraph	Motor Graphic	
Signal Input	Signal Output	Signal Type
DrivePos		Sig Thev
DriveNeg		Sig Thev
	Shaft	Sig Force
	Case	Sig Force
	ShaftAngle	Sig Double
	ShaftAngleAsVoltage	Sig Voltage
	ShaftVelocityAsVoltage	Sig Voltage
	ShaftAccelAsVoltage	Sig Voltage
	MotorCurrentAsVoltage	Sig Voltage
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	<pre>double velocity; double position; ThevSig drivepos_in; ThevSig drivepos_out; ThevSig driveneg_in; ThevSig driveneg_out; ForceSig shaftforce_in; ForceSig shaftforce_out; ForceSig caseforce_in;</pre>	



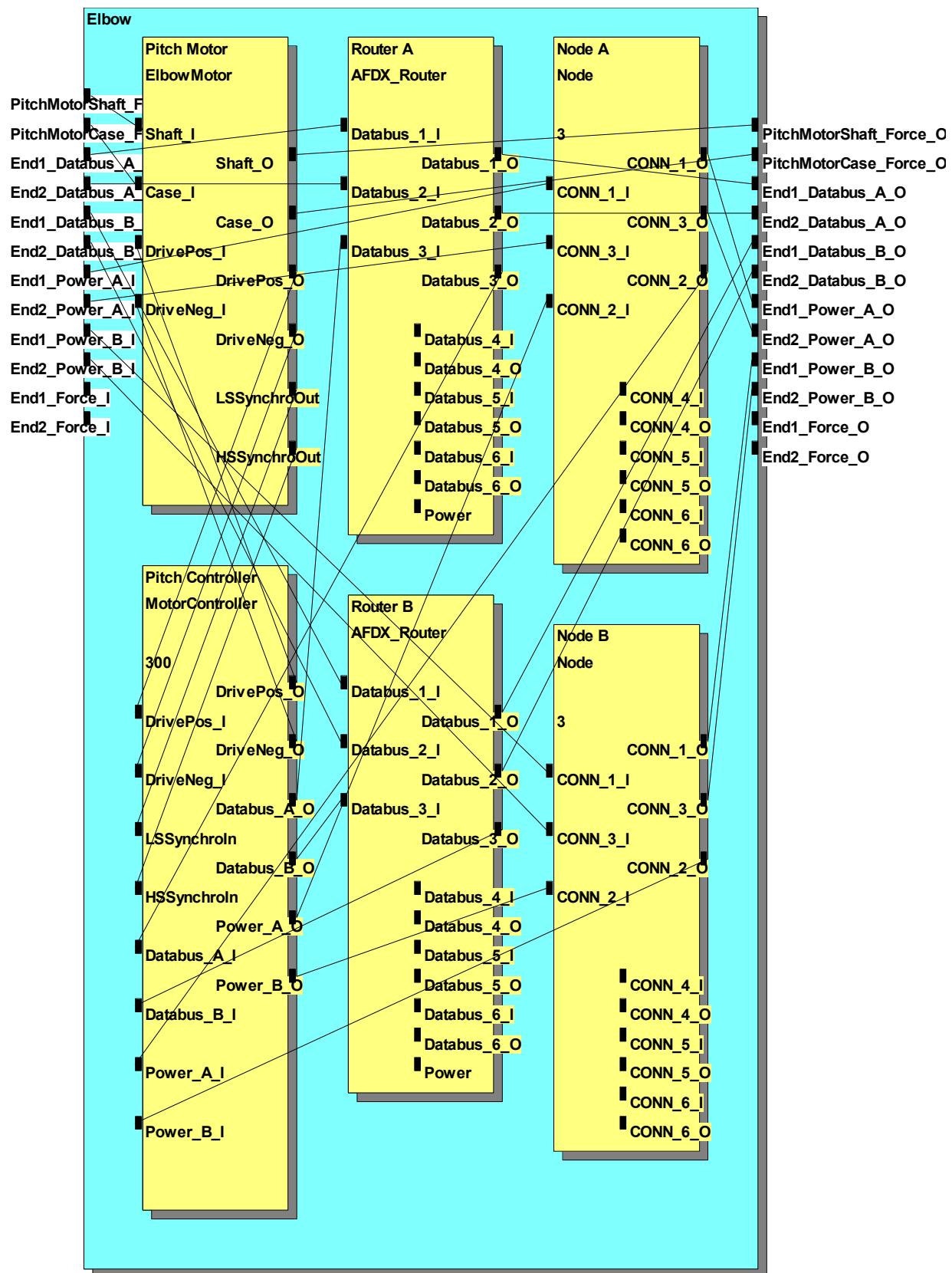
	<pre>ForceSig caseforce_out; //MechanicalPart mechPart; // motor variables double emotor, inductance, current, backemf, torque, mag_torque; double radvel, torque_const, backemf_const, inertia; double const_friction, viscous_friction, energy, power; double current_dot, radvel_dot, gear_ratio; double flywheel_mass, flywheel_radius, flywheel_inertia; double rotor_mass, rotor_radius, rotor_inertia; double case_radius; double deltime; double arm_resistance, switch_resistance; double rated_voltage, no_load_vel, blocked_torque; double lvolts; double shaft_angle; int ysize; int cnt; double time; double pulse_low_time; double pulse_high_time; double switch_freq, switch_period, on_duty_cycle, on_time, off_time; int on_flag; short xtime; ParticleGroup *motorCase; ParticleGroup *motorShaft;</pre>
Requirements	<ol style="list-style-type: none">1. Receive Positive and Negative DC Voltages and apply to motor coil.2. Integrate motor inductive voltage.3. Integrate motor inertia.4. Model reverse emf.5. Apply torque to connected shaft.
Limitations	Uses and ideal current to torque relationship. Could be improved by simulating individual quadrants.





3.5.4.6 Elbow

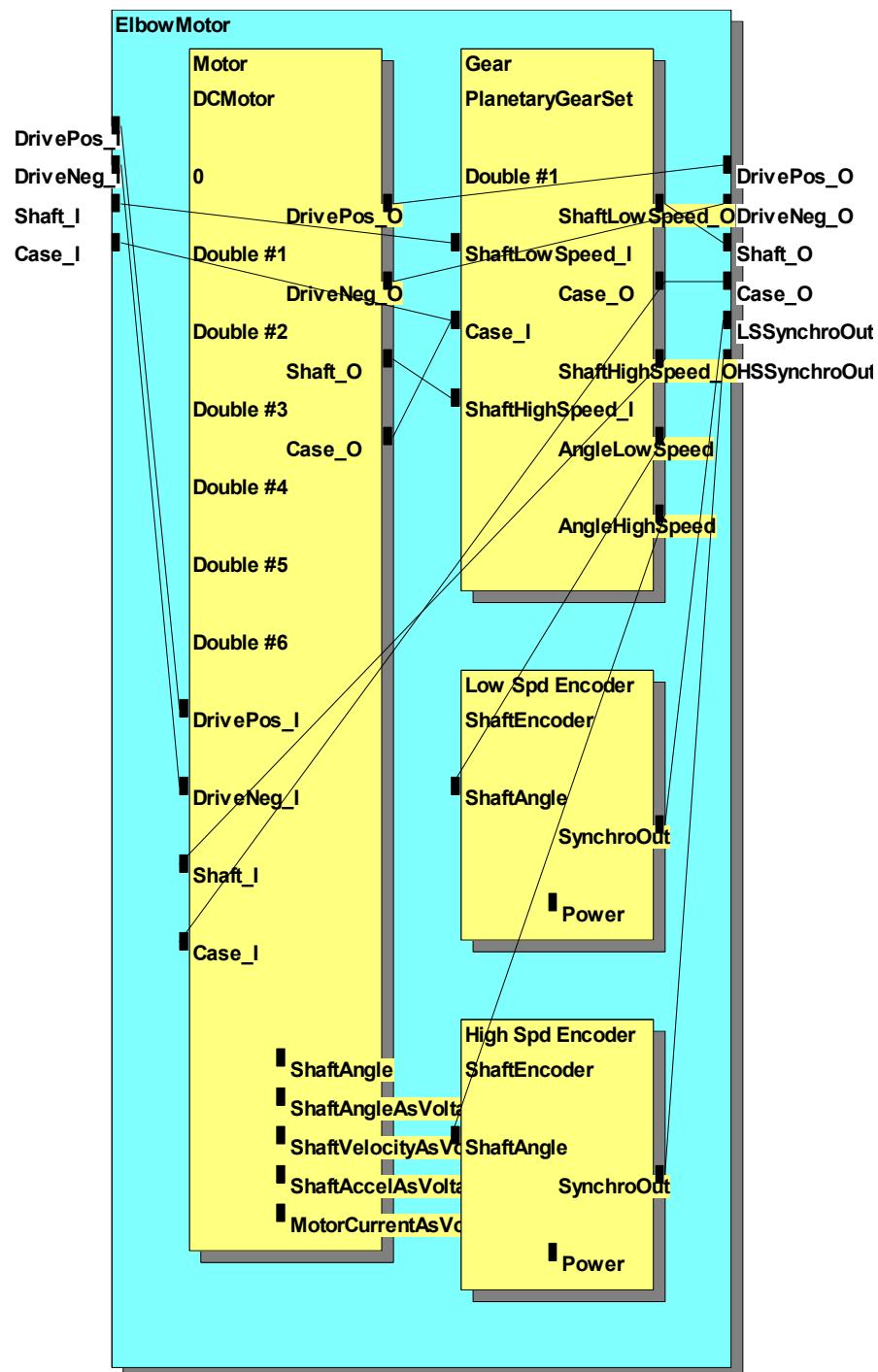
Class Name	Elbow	
Sub-Part Class	Sub-Part Instance	
ElbowMotor	Pitch Motor	
MotorController	Pitch Controller	
AFDX_Router	Router A Router B	
Node	Node A Node B	
Signal Input	Signal Output	Signal Type
End1_Databus_A_I	End1_Databus_A_O	Sig AFDX
End1_Databus_B_I	End1_Databus_B_O	Sig AFDX
End2_Databus_A_I	End2_Databus_A_O	Sig AFDX
End2_Databus_B_I	End2_Databus_B_O	Sig AFDX
End1_Force_I	End1_Force_O	Sig Force
End2_Force_I	End2_Force_O	Sig Force
PitchMotorShaft_Force_I	PitchMotorShaft_Force_O	Sig Force
PitchMotorCase_Force_I	PitchMotorCase_Force_O	Sig Force
End1_Power_A		Sig Thev
End1_Power_B		Sig Thev
End2_Power_A		Sig Thev
End2_Power_B		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.7 Elbow Motor

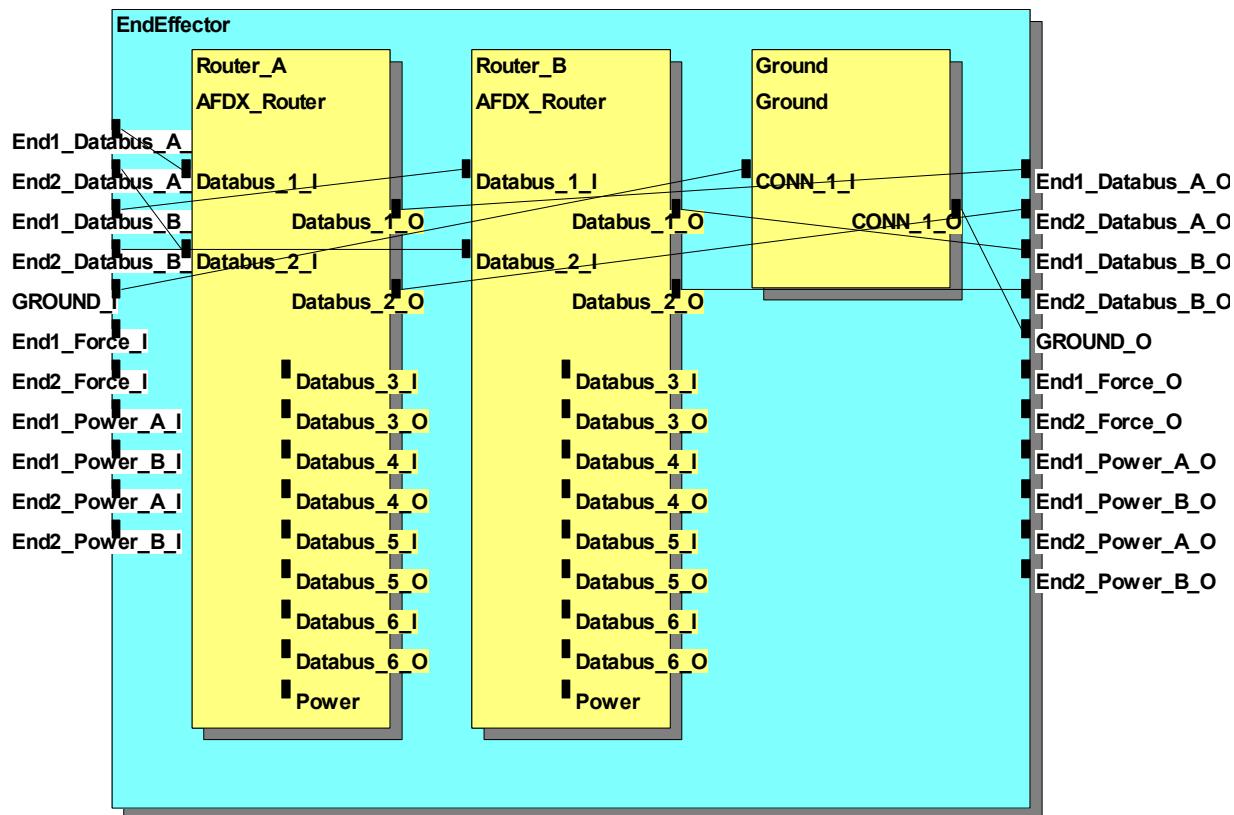
Class Name	ElbowMotor	
Sub-Part Class	Sub-Part Instance	
DCMotor	Motor	
PlanetaryGearSet	Gear	
Shaft Encoder	Low Speed Encoder	
	High Speed Encoder	
Signal Input	Signal Output	Signal Type
DrivePos		Sig Thev
DriveNeg		Sig Thev
Shaft		Sig Force
Case		Sig Force
	LSSynchroOut	Sig Synchro
	HSSynchroOut	Sig Synchro
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.8 End Effector

Class Name	End Effector	
Sub-Part Class	Sub-Part Instance	
AFDX_Router	Router A Router B	
Ground	Ground	
Signal Input	Signal Output	Signal Type
End1_Databus_A_I	End1_Databus_A_O	Sig AFDX
End1_Databus_B_I	End1_Databus_B_O	Sig AFDX
End2_Databus_A_I	End2_Databus_A_O	Sig AFDX
End2_Databus_B_I	End2_Databus_B_O	Sig AFDX
End1_Force_I	End1_Force_O	Sig Force
End2_Force_I	End2_Force_O	Sig Force
End1_Power_A		Sig Thev
End1_Power_B		Sig Thev
End2_Power_A		Sig Thev
End2_Power_B		Sig Thev
GROUND		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	



3.5.4.9 Ground

Class Name	Ground	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
CONN_1_I		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Model a Thevenin Equivalent zero ohms & zero volts.	
Limitations	None	





3.5.4.10 Image Sensor

The Image Sensor converts the image captured by the camera lens into a digital bit map, and provides an interface with the AFDX bus for sending the data to the RMS Computer. [Table 14](#) and [Table 15](#) list the AFDX command and response data formats used for communicating with the Image Sensor part. Command and data words are 32-bit values. All message packets have a maximum of 16 data words (data[0] – data[15]).

There are three basic message types used by the RMS Computer to communicate with all AFDX devices:

- a) **Query Status – All:** Requests the status of all AFDX devices;
- b) **Query Response:** Requests data of a specific AFDX device;
- c) **Execute Command:** Issues a command for a specific AFDX device to execute.

AFDX messages from the RMS Computer to the Image Sensors take the form shown in [Table 14](#).

Table 14: AFDX Messages from RMS Computer to an Image Sensor

Message Type	AFDX Dest. Addx	Src Addx Data[0]	Data Type Data[1]
1 = Query Status - All	Unused	0x0000	Unused
2 = Query Response	Device Addx	0x0000	1 = Status
			9 = CamImage
3 = Execute Command	Device Addx	0x0000	2 = Self Test

The Image Sensor sends messages only in response to commands from a controlling device (e.g. RMS Computer). The format of the Image Sensor response messages is shown in [Table 15](#). The Message Type field contains the code for the RMS Computer message for which the response has been generated.

Table 15: AFDX Messages from an Image Sensor to the RMS Computer

Message Type	Dest. Addx	Src. Addx Data[0]	Data Type Data[1]	Status Wd Data[2]	Pointer Flag	Num Bytes	Data Pointer
1 = Query Status - All	0x0000	Device Addx	1 = Status	16-bit status word	FALSE	Unused	Unused
2 = Query Response	0x0000	Device Addx	1 = Status	16-bit status word	FALSE	Unused	Unused
			9 = Camera Image	16-bit status word	TRUE	307,200	Pointer to Image Bitmap
3 = Execute Command	0x0000	Device Addx	1 = Status	16-bit status word	FALSE	Unused	Unused

[Table 16](#) gives an example of the code used to send AFDX messages to the Image Sensor part. Please refer to the full RMS Computer ES code listing for more detail and examples.

Table 16: Code Example for Sending Messages to an Image Sensor

```
// Send AFDX commands to query all camera image sensors
for(cam = 0; cam < NumCameras; cam++) {

    afdxsigout.msgType = AFDXSig::QueryResponse; // Command
    afdxsigout.address = imageSensorAFDXAddr[cam]; // Dest addr
    afdxsigout.data[1] = CamImage; // Data Type
    SendSignal(Databus_A_O, &afdxsigout); // Send Command
}
```



[Table 17](#) gives an example of the code used to read data from the Image Sensor part. Please refer to the RMS Computer ES code (RMSComputer.cpp) for more detail and examples.

Table 17: ES Code Example for Receiving Messages From an Image Sensor

```

case Databus_A_I: // Handle all incoming AFDX data
    afdxsign = *((AFDXSig *)data);

    // Handle AFDX inputs from camera motor controllers & image sensors
    for(cam = 0; cam < NumCameras; cam++) {

        // Check for video data from camera image sensors
        if(afdxsign.data[0] == imageSensorAFDXAddr[cam]) { // Address match?
            switch(afdxsign.data[1]) { // What type of data is it?

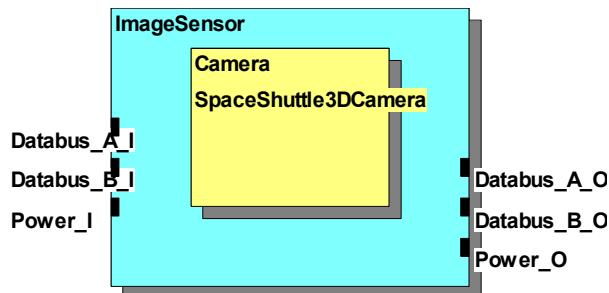
                case Status: // Status data
                    imgSensorStatus[cam] = afdxsign.data[2]; // Store status data
                    break; //End Status

                case CamImage: // Camera image bitmap
                    camBitMap[cam] = (unsigned char *)(afdxsign.imgBitMapPtr);
                    imgSensorStatus[cam] = afdxsign.data[2]; // Store status data
                    camBitMapFresh[cam] = TRUE;
                    break; //End CamImage
            }
        }
    }
}

```

The Image Sensor part summary table and part block diagram are given below:

Class Name	ImageSensor	
Sub-Part Class	Sub-Part Instance	
SpaceShuttle3DCamera	Camera	
Signal Input	Signal Output	Signal Type
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Power		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	void ImageSensor::SetCameraPosition(double x, double y, double z) void ImageSensor::SetCameraAngle(double yaw, double pitch, double zoom)	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	AFDXSig afdxout; unsigned char camerabm[320 * 240 * 3 + 1];	
Requirements	Send Image Data in response to AFDX commands	
Limitations	None	



3.5.4.11 Lamp

While no functionality has been implemented, the Camera Lamp is intended to provide adjustable illumination on the subject viewed by the cameras. The Lamp part receives command data from the RMS Computer via the AFDX bus. [Table 18](#) lists the AFDX command data formats used for communicating with the Lamp part. Command and data words are 32-bit values. All message packets have a maximum of 16 data words (data[0] – data[15]).

There are three basic message types used by the RMS Computer to communicate with all AFDX devices:

- Query Status – All:** Requests the status of all AFDX devices;
- Query Response:** Requests data of a specific AFDX device;
- Execute Command:** Issues a command for a specific AFDX device to execute.

AFDX messages from the RMS Computer to the Camera Lamps take the form shown in [Table 18](#).

Table 18: AFDX Messages from RMS Computer to a Camera Lamp

Message Type	AFDX Dest. Addx	Src Addx Data[0]	Data Type Data[1]	Data Type Data[2]
3 = Execute Command	Device Addx	0x0000	8 = Illumination	Lamp Intensity (0-100% (Integer))

The Camera Lamp sends no messages in response to commands from a controlling device (e.g. RMS Computer).

[Table 19](#) gives an example of the code used to send AFDX messages to the Camera Lamp part. Please refer to the full RMS Computer ES code listing for more detail and examples.

Table 19: ES Code Example for Sending Messages to a Camera Lamp

```

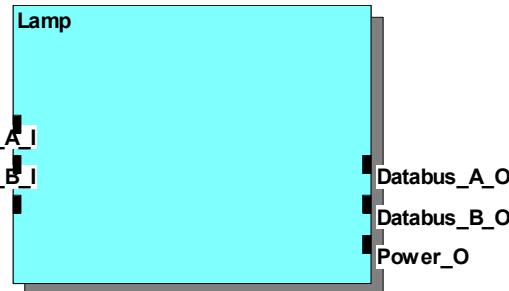
// Send AFDX commands to query all camera image sensors
for(cam = 0; cam < NumCameras; cam++) {

    // Send command to active camera lamp
    if(chgLampInten[activeCam]) {
        afdxsigout.msgType = AFDXSig::ExecuteCmd;
        afdxsigout.address = camLampAFDXAddr[activeCam];
        afdxsigout.data[1] = Illumination; // Data type = Intensity
        afdxsigout.data[2] = lampIntensity[activeCam]; // Intensity Pct
        SendSignal(Databus_A_O, &afdxsigout); // Issue command

        chgLampInten[activeCam] = FALSE;
    }
}
  
```



Class Name	Lamp	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Power_I		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	Tbd	
Requirements	Control lamp brightness based on AFDX commands	
Limitations	None	

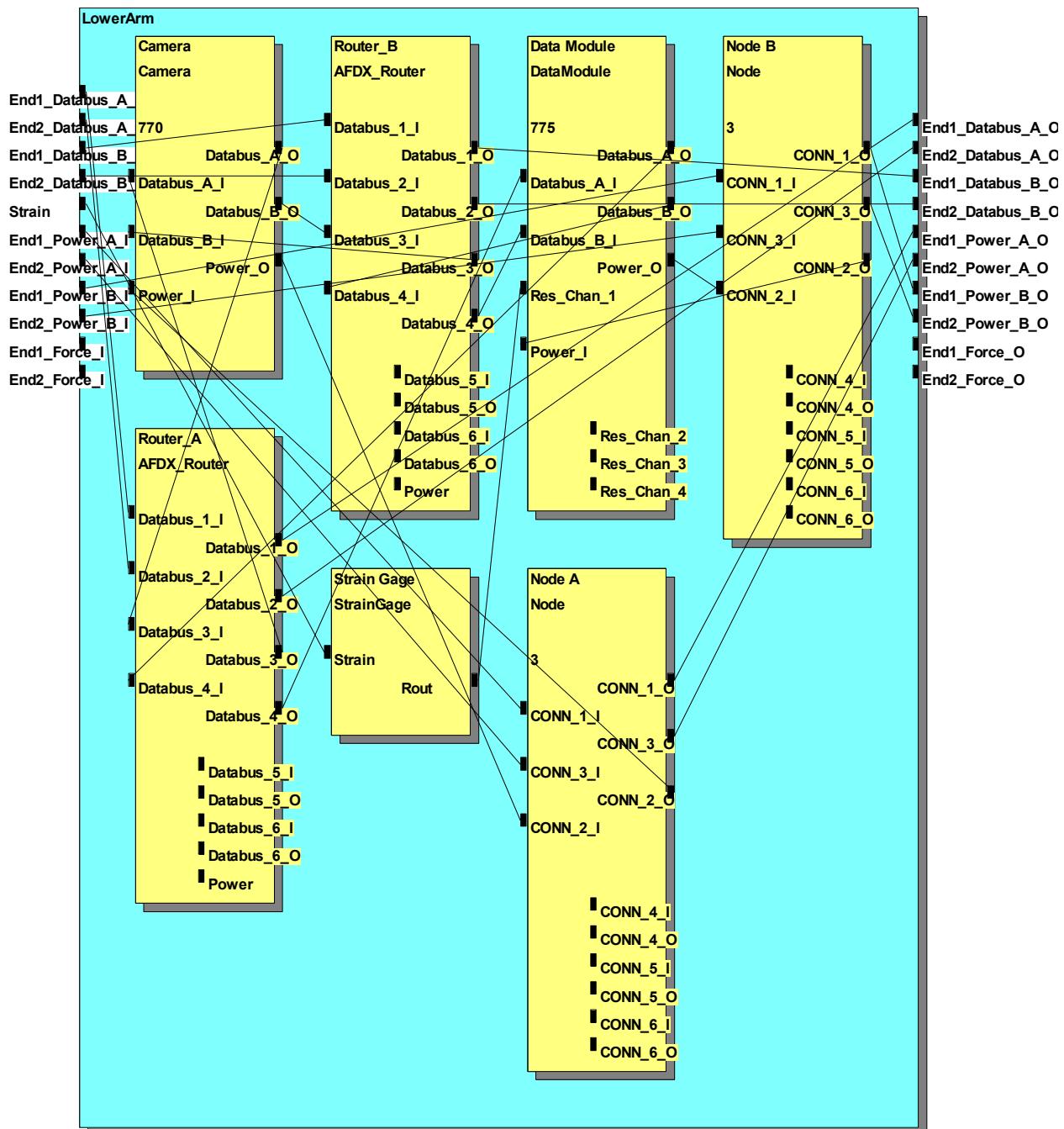


3.5.4.12 Lower Arm

Class Name	LowerArm	
Sub-Part Class	Sub-Part Instance	
Camera	Camera	
AFDX_Router	Router_A Router_B	
StrainGage	Strain Gage	
DataModule	Data Module	
Node	Node A Node B	
Signal Input	Signal Output	Signal Type
End1_Databus_A_I	End1_Databus_A_O	Sig AFDX
End1_Databus_B_I	End1_Databus_B_O	Sig AFDX
End2_Databus_A_I	End2_Databus_A_O	Sig AFDX
End2_Databus_B_I	End2_Databus_B_O	Sig AFDX
End1_Force_I	End1_Force_O	Sig Force
End2_Force_I	End2_Force_O	Sig Force
End1_Power_A		Sig Thev
End1_Power_B		Sig Thev



End2_Power_A		Sig Thev
End2_Power_B		Sig Thev
Strain		Sig Double
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	Inertial Model variables	
Requirements	Send strain signals to strain gage	
Limitations	None	





3.5.4.13 Motor Controller

The Motor Controller part is responsible for controlling all DC motors used throughout the SRMS, and communicating with the RMS Computer via the AFDX bus. [Table 20](#) and [Table 21](#) list the AFDX command and response data formats used for communicating with the Motor Controller part. As with the SPI bus, command and data words are 16 bit values and the maximum packet size is 16 words.

There are three basic message types used by the RMS Computer to communicate with all AFDX devices:

- a) **Query Status – All:** Requests the status of all AFDX devices;
- b) **Query Response:** Requests data of a specific AFDX device;
- c) **Execute Command:** Issues a command for a specific AFDX device to execute.

AFDX messages from the RMS Computer to the Motor Controllers take the form shown in [Table 20](#).

Table 20: AFDX Messages from RMS Computer to a Motor Controller

Message Type	AFDX Dest. Addx	Src Addx Data[0]	Data Type Data[1]	Data Wd Data[2]
1 = Query Status - All	Unused	0x0000	Unused	Unused
2 = Query Response	Device Addx	0x0000	1 = Status 4 = Motor Velocity 5 = Motor Angle 6 = Joint Angle (degrees)	Unused
3 = Execute Command	Device Addx	0x0000	2 = Self Test 4 = Motor Velocity (RPM)	16-bit Integer

The Motor Controller sends messages only in response to commands from a controlling device (e.g. RMS Computer). The format of the Motor Controller response messages is shown in [Table 21](#). The Message Type field contains the code for the RMS Computer message for which the response has been generated.

Table 21: AFDX Messages from Motor Controller to the RMS Computer

Message Type	AFDX Dest. Addx	Src. Addx Data[0]	Data Type Data[1]	Status Wd Data[2]	Data Wd Data[3]	Data Wd Data[4]
1 = Query Status - All	0x0000	Device Addx	1 = Status	16-bit status word	Unused	Unused
2 = Query Response	0x0000	Device Addx	1 = Status	16-bit status word	Unused	Unused
			4 = Motor Velocity	16-bit status word	16-bit Integer	Unused
			5 = Motor Angle	16-bit status word	0x0000	Unused
			6 = Joint Angle (degrees)	16-bit status word	Joint Angle Integer	Joint Angle Fraction * 4096
3 = Execute Command	0x0000	Device Addx	1 = Status	16-bit status word	Unused	Unused

[Table 22](#) gives an example of the code used to send AFDX messages to the Motor Controller part. Please refer to the full RMS Computer ES code listing for more detail and examples.

**Table 22: Code Example for Sending Messages to a Motor Controller**

```
// Build & send command to motor
afdxsigout.msgType = AFDXSig::ExecuteCmd;
afdxsigout.address = rmaMotorAFDXAddr[joint];
afdxsigout.data[1] = MotorVelocity; // Data type = Motor Velocity
afdxsigout.data[2] = motorSpeedCmd[joint]; // Motor velocity (joint speed = motor speed / 500)
SendSignal(Databus_A_O, &afdxsigout); // Issue command
```

[Table 23](#) gives an example of the code used to read data from the RMS Control Panel part. Please refer to the RMS Computer ES code (RMSComputer.cpp) for more detail and examples.

Table 23: Code Example for Receiving Messages From a Motor Controller

```
case Databus_A_I: // Handle all incoming AFDX data
    afdxsignin = *((AFDXSig *)data);

    // Handle AFDX inputs from RMA motor controllers
    for(joint = 0; joint < NumRMAJoints; joint++) {
        if(afdxsignin.data[0] == rmaMotorAFDXAddr[joint]) { // Address match?
            switch(afdxsignin.data[1]) { // What type of data is it?

                case Status: // Status data
                    rmaJointStatus[joint] = afdxsignin.data[2];
                    break; //End Status

                case JointAngle: // Joint angle data
                    rmaJointStatus[joint] = afdxsignin.data[2];
                    measAngle[joint] = (int)afdxsignin.data[3];
                    measAngleFract[joint] = (int)afdxsignin.data[4];
                    freshAngleData[joint] = TRUE;
                    jointAngleRqstd[joint] = FALSE;
                    break; //End JointAngle

                case MotorVelocity: // Motor shaft velocity
                    // Data not currently used
                    break; // End MotorVelocity

                case MotorAngle: // Motor shaft position
                    // Data not currently used
                    break; // End MotorAngle
            }
        }
    }
```

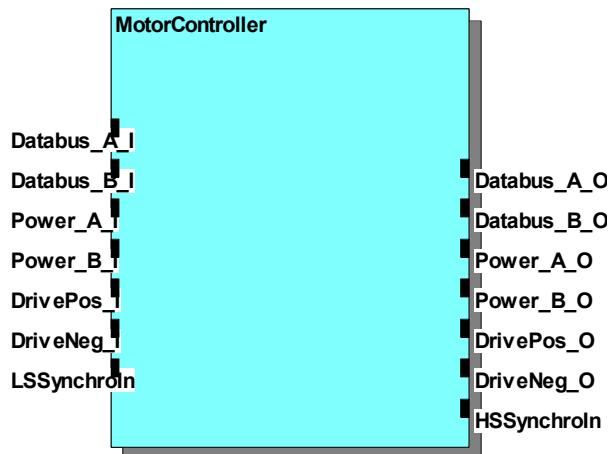


The Motor Controller part summary table and part block diagram are given below:

Class Name	MotorController	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
DrivePos		Sig Thev
DriveNeg		Sig Thev
LSSynchroIn		Sig Synchro
HSSynchroIn		Sig Synchro
Power_A		Sig Thev
Power_B		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	<pre>AFDXSig afdxout; double velocity; enum { PCM_EVENT, POLL_EVENT, }; AddrDataGroup *adgrptr; BOOL power_on; BOOL opt1_state; BOOL opt2_state; int optstate; int newoptstate; int cw; int cew; short pcm_percent; short pcm_us_period; short ton; short toff; BOOL pos_direction; BOOL onState; BoolSig bfalse; BoolSig btrue; short measRpm; short rpmErr;</pre>	



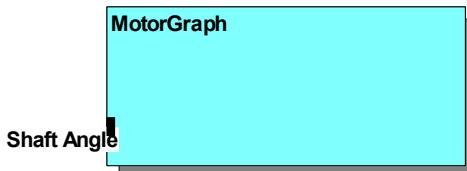
	<pre>short measAccel; short lastMeasRpm; short vMax; short aMax; short setPointRpm; short setPointAccel; short pcmPercent; short lastVelSet; short Kp; short Ki; short Kd; short dcErr; short pTerm; short sumErr; short iTerm; short dTerm; double motorShaftAngle; double motorShaftAngleLast; double lowSpeedAngle;</pre>
Requirements	Use a PID algorithm to control DC motor speed. Output pulse code modulated voltages to DC motor. Receive & respond to AFDX commands.
Limitations	None





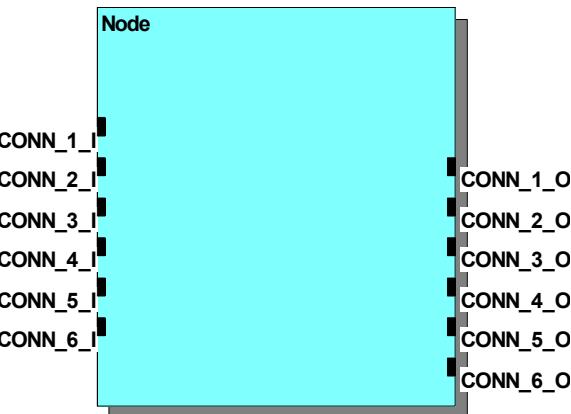
3.5.4.14 Motor Graph

Class Name	MotorGraph	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
Shaft Angle		Sig Double
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Display 2-D graphic of motor rotation	
Limitations	None	



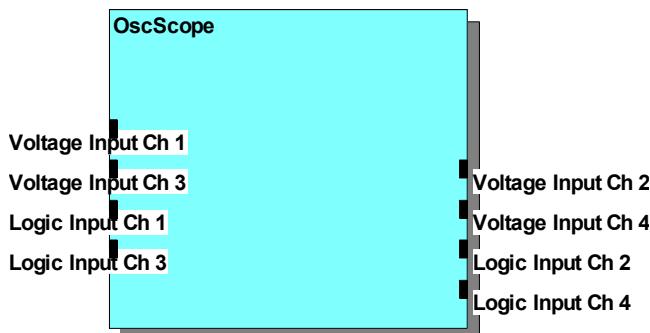
3.5.4.15 Node

Class Name	Node	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
CONN_1		Sig Thev
CONN_2		Sig Thev
CONN_3		Sig Thev
CONN_4		Sig Thev
CONN_5		Sig Thev
CONN_6		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Join analog signals together	
Limitations	None	



3.5.4.16 OscScope

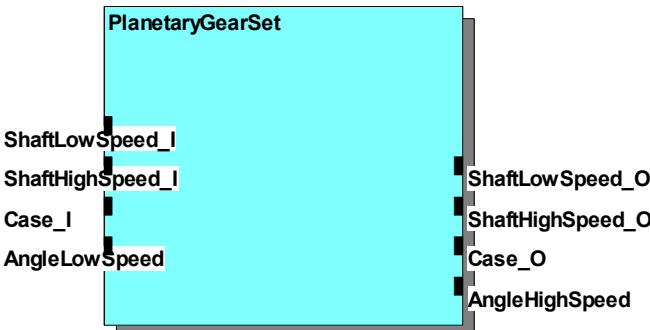
Class Name	OscScope	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
Voltage Input Ch 1		Sig Voltage
Voltage Input Ch 2		Sig Voltage
Voltage Input Ch 3		Sig Voltage
Voltage Input Ch 4		Sig Voltage
Logic Input Ch 1		Sig Bool
Logic Input Ch 2		Sig Bool
Logic Input Ch 3		Sig Bool
Logic Input Ch 4		Sig Bool
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Display up to 4 voltage waveforms	
Limitations	None	





3.5.4.17 Planetary Gear Set

Class Name	PlanetaryGearSet	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
ShaftLowSpeed_I	ShaftLowSpeed_O	Sig Force
ShaftHighSpeed_I	ShaftHighSpeed_O	Sig Force
Case_I	Case_O	Sig Force
	AngleLowSpeed	Sig Double
	AngleHighSpeed	Sig Double
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Convert one shaft speed to another	
Limitations	None	

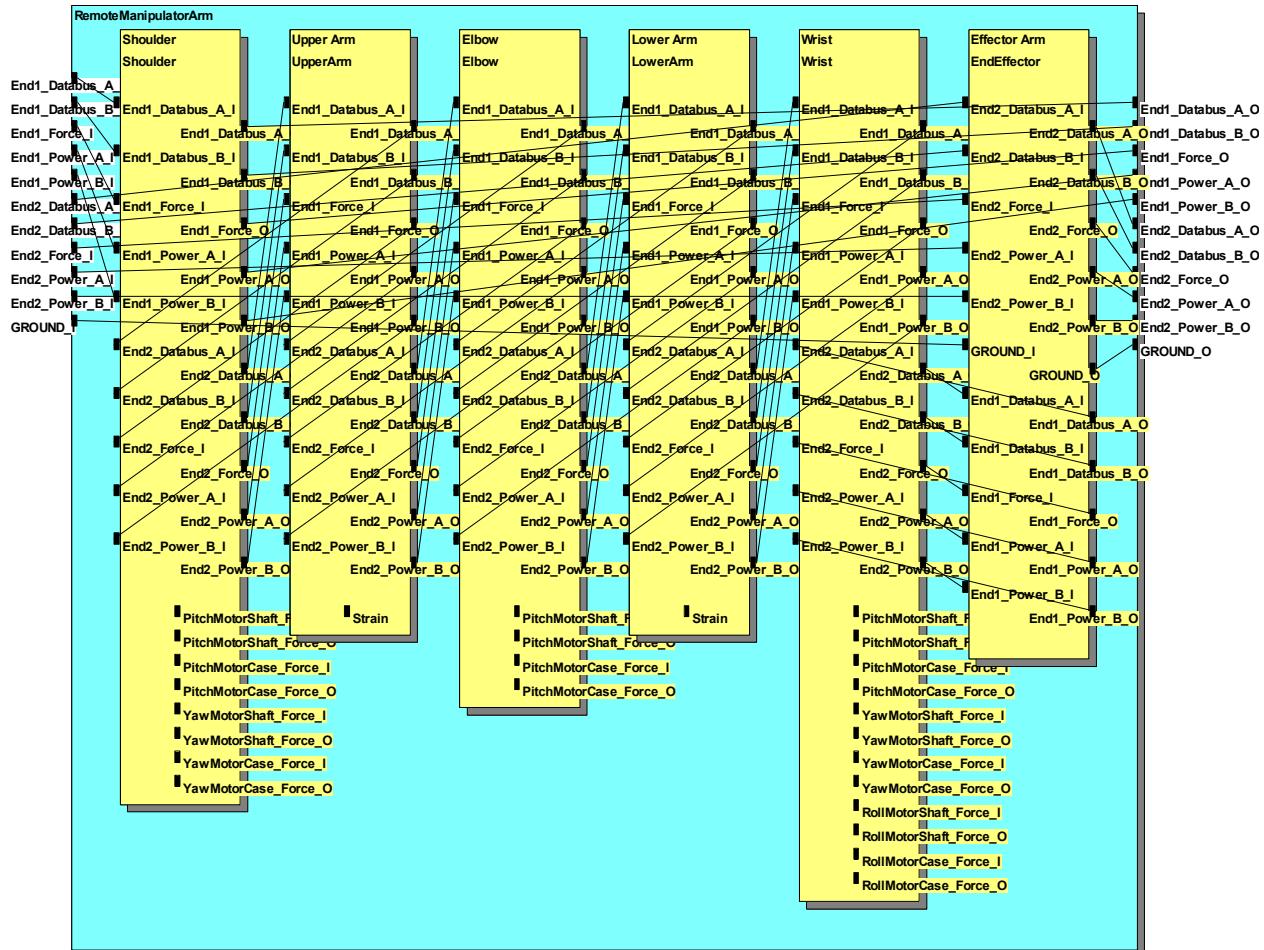


3.5.4.18 Remote Manipulator Arm

Class Name	RemoteManipulatorArm	
Sub-Part Class	Sub-Part Instance	
Shoulder	Shoulder	
UpperArm	UpperArm	
Elbow	Elbow	
LowerArm	LowerArm	
Wrist	Wrist	
EffectorArm	EndEffector	
Signal Input	Signal Output	Signal Type
End1_Databus_A_I	End1_Databus_A_O	Sig AFDX
End1_Databus_B_I	End1_Databus_B_O	Sig AFDX
End2_Databus_A_I	End2_Databus_A_O	Sig AFDX
End2_Databus_B_I	End2_Databus_B_O	Sig AFDX
End1_Force_I	End1_Force_O	Sig Force
End2_Force_I	End2_Force_O	Sig Force



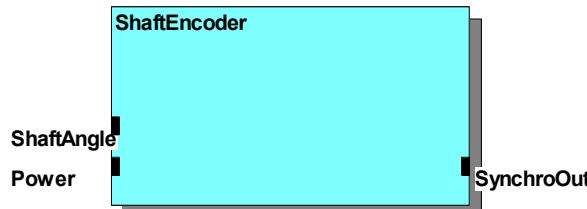
End1_Power_A		Sig_Thev
End1_Power_B		Sig_Thev
End2_Power_A		Sig_Thev
End2_Power_B		Sig_Thev
Ground		Sig_Double
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.19 Shaft Encoder

Class Name	ShaftEncoder	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
ShaftAngle		Sig Double
	SynchroOut	Sig Synchro
Power		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Convert shaft angle voltage to synchro signal	
Limitations	None	





3.5.4.20 Shoulder

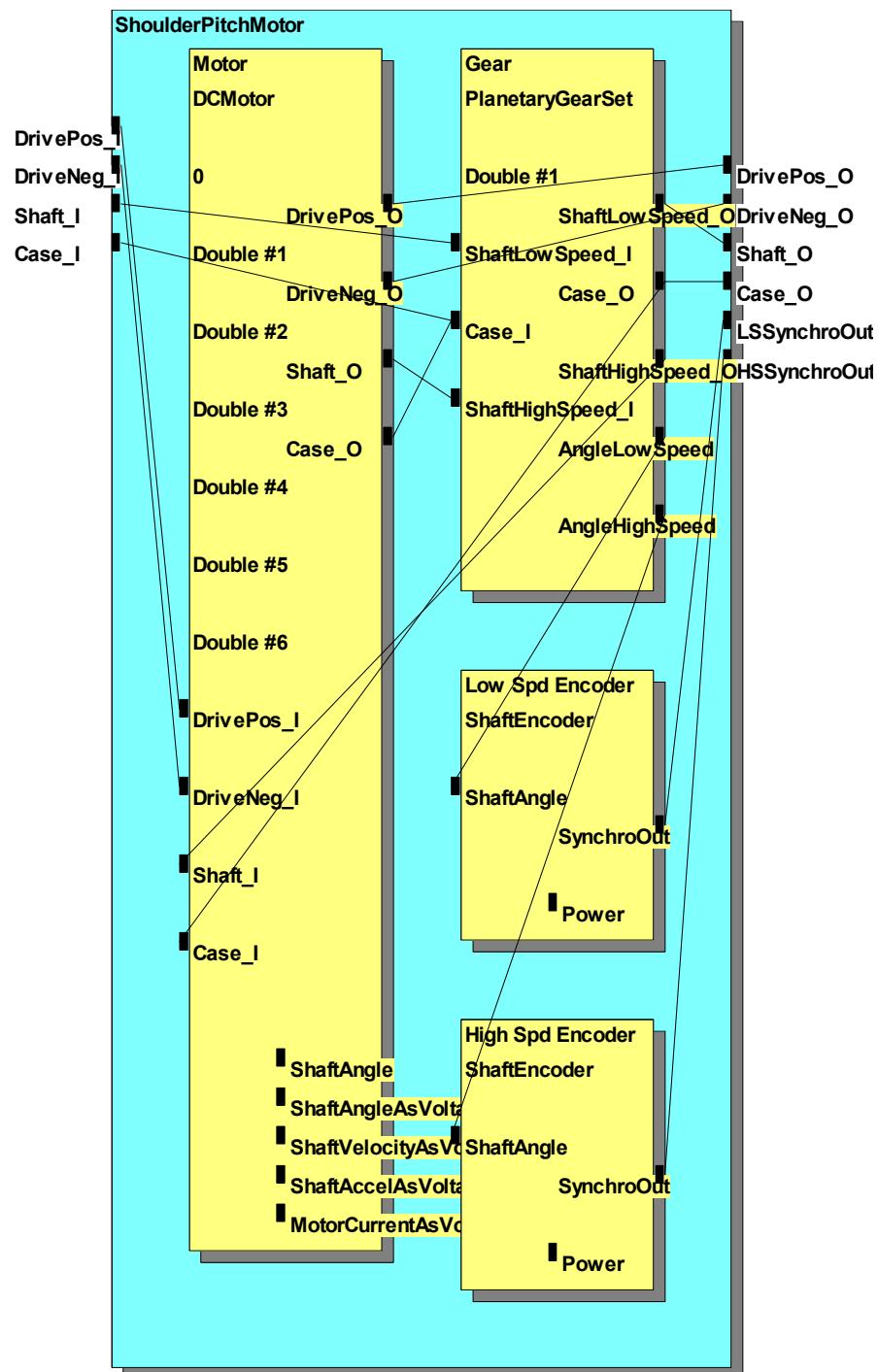
Class Name	Shoulder	
Sub-Part Class	Sub-Part Instance	
ShoulderPitchMotor	Pitch Motor	
ShoulderYawMotor	Yaw Motor	
MotorController	Pitch Controller Yaw Controller	
AFDX_Router	Router A Router B	
Node	Node A Node B	
Signal Input	Signal Output	Signal Type
End1_Databus_A_I	End1_Databus_A_O	Sig AFDX
End1_Databus_B_I	End1_Databus_B_O	Sig AFDX
End1_Force_I	End1_Force_O	Sig Force
End1_Power_A		Sig Thev
End1_Power_B		Sig Thev
End2_Databus_A_I	End2_Databus_A_O	Sig AFDX
End2_Databus_B_I	End2_Databus_B_O	Sig AFDX
End2_Force_I	End2_Force_O	Sig Force
	End2_Power_A	Sig Thev
	End2_Power_B	Sig Thev
PitchMotorShaft_Force_I	PitchMotorShaft_Force_O	Sig Force
PitchMotorCase_Force_I	PitchMotorCase_Force_O	Sig Force
YawMotorShaft_Force_I	YawMotorShaft_Force_O	Sig Force
YawMotorCase_Force_I	YawMotorCase_Force_O	Sig Force
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.21 Shoulder Pitch Motor

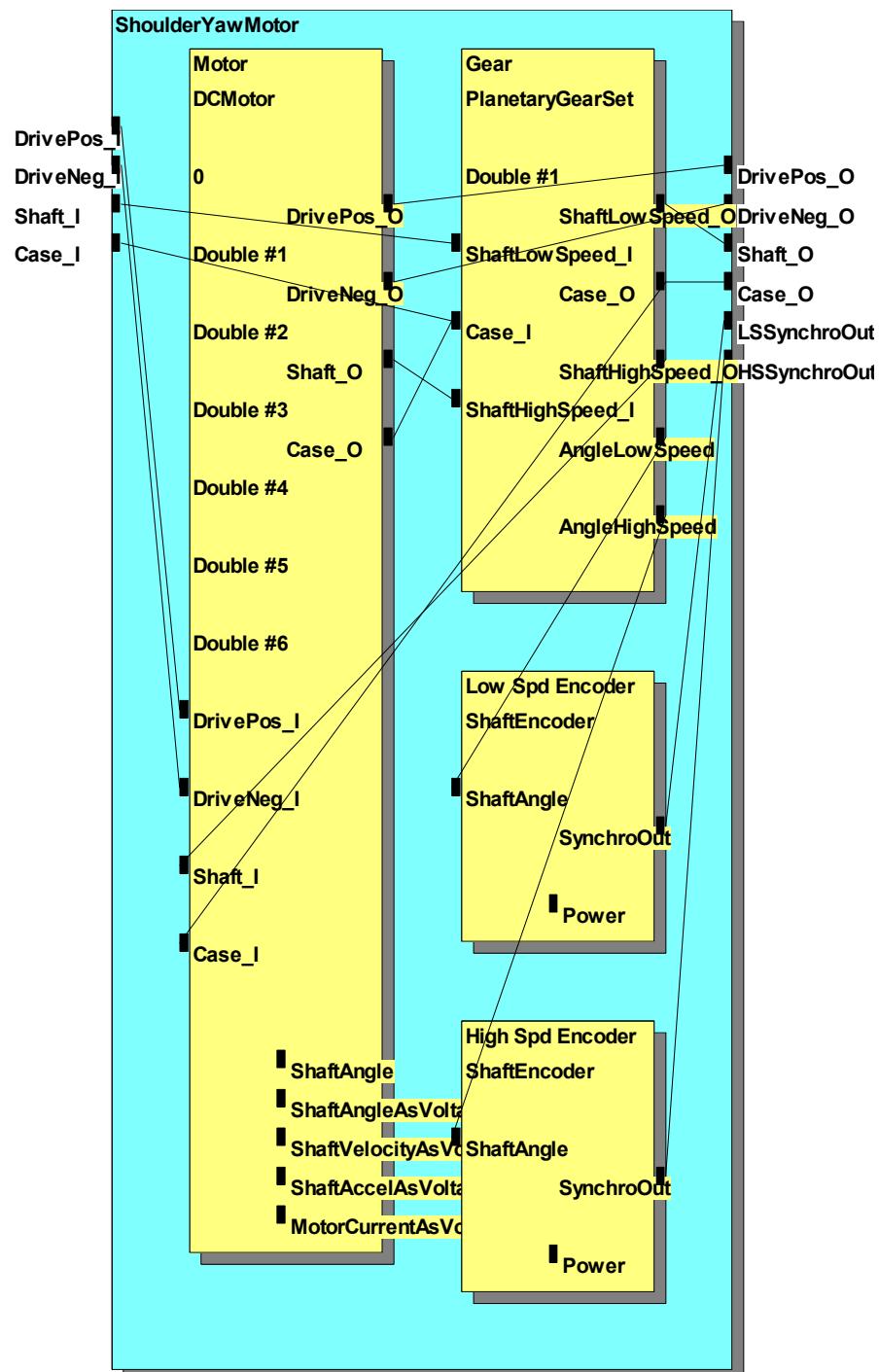
Class Name	ShoulderPitchMotor	
Sub-Part Class	Sub-Part Instance	
DCMotor	Motor	
PlanetaryGearSet	Gear	
Shaft Encoder	Low Speed Encoder	
	High Speed Encoder	
Signal Input	Signal Output	Signal Type
DrivePos		Sig Thev
DriveNeg		Sig Thev
Shaft		Sig Force
Case		Sig Force
	LSSynchroOut	Sig Synchro
	HSSynchroOut	Sig Synchro
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.22 Shoulder Yaw Motor

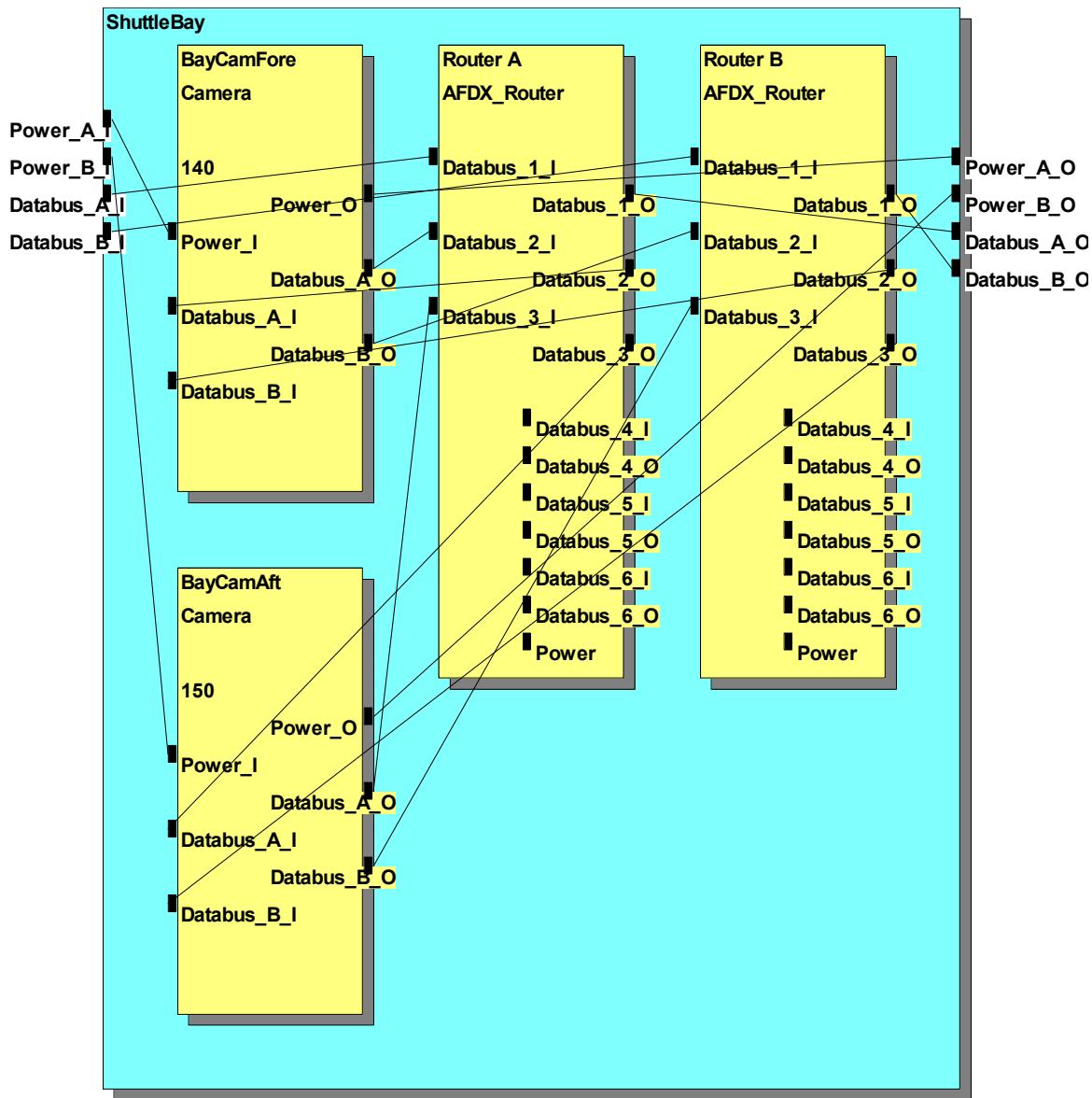
Class Name	ShoulderYawMotor	
Sub-Part Class	Sub-Part Instance	
DCMotor	Motor	
PlanetaryGearSet	Gear	
Shaft Encoder	Low Speed Encoder	
	High Speed Encoder	
Signal Input	Signal Output	Signal Type
DrivePos		Sig Thev
DriveNeg		Sig Thev
Shaft		Sig Force
Case		Sig Force
	LSSynchroOut	Sig Synchro
	HSSynchroOut	Sig Synchro
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.23 Shuttle Bay

Class Name	ShuttleBay	
Sub-Part Class	Sub-Part Instance	
Camera	BayCamAft BayCamFore	
AFDX_Router	Router A Router B	
Signal Input	Signal Output	Signal Type
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Power_A		Sig Thev
Power_B		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	



3.5.4.24 Space Shuttle 3D camera

Class Name	SpaceShuttle3Dcamera	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
None		
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	

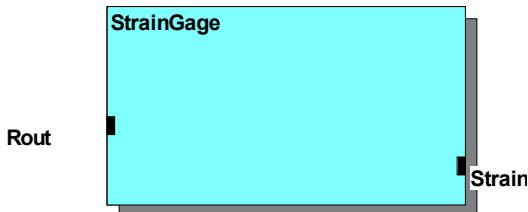


Internal State Variables	None
Requirements	Tbs
Limitations	None



3.5.4.25 Strain Gage

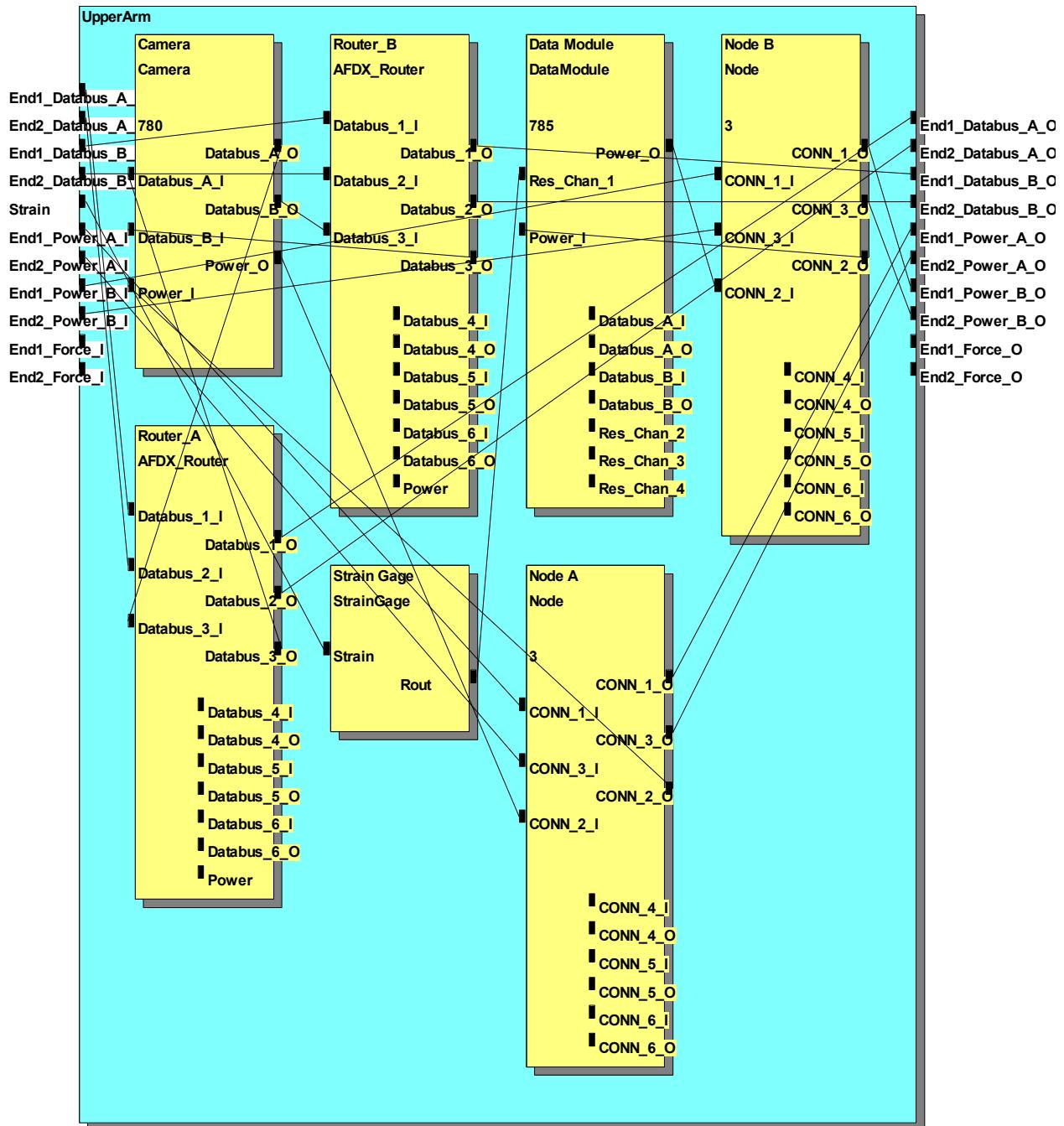
Class Name	StrainGage	
Sub-Part Class	Sub-Part Instance	
None		
Signal Input	Signal Output	Signal Type
	Rout	Sig Thev
Strain		Sig Force
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Convert strain force into resistance	
Limitations	None	





3.5.4.26 Upper Arm

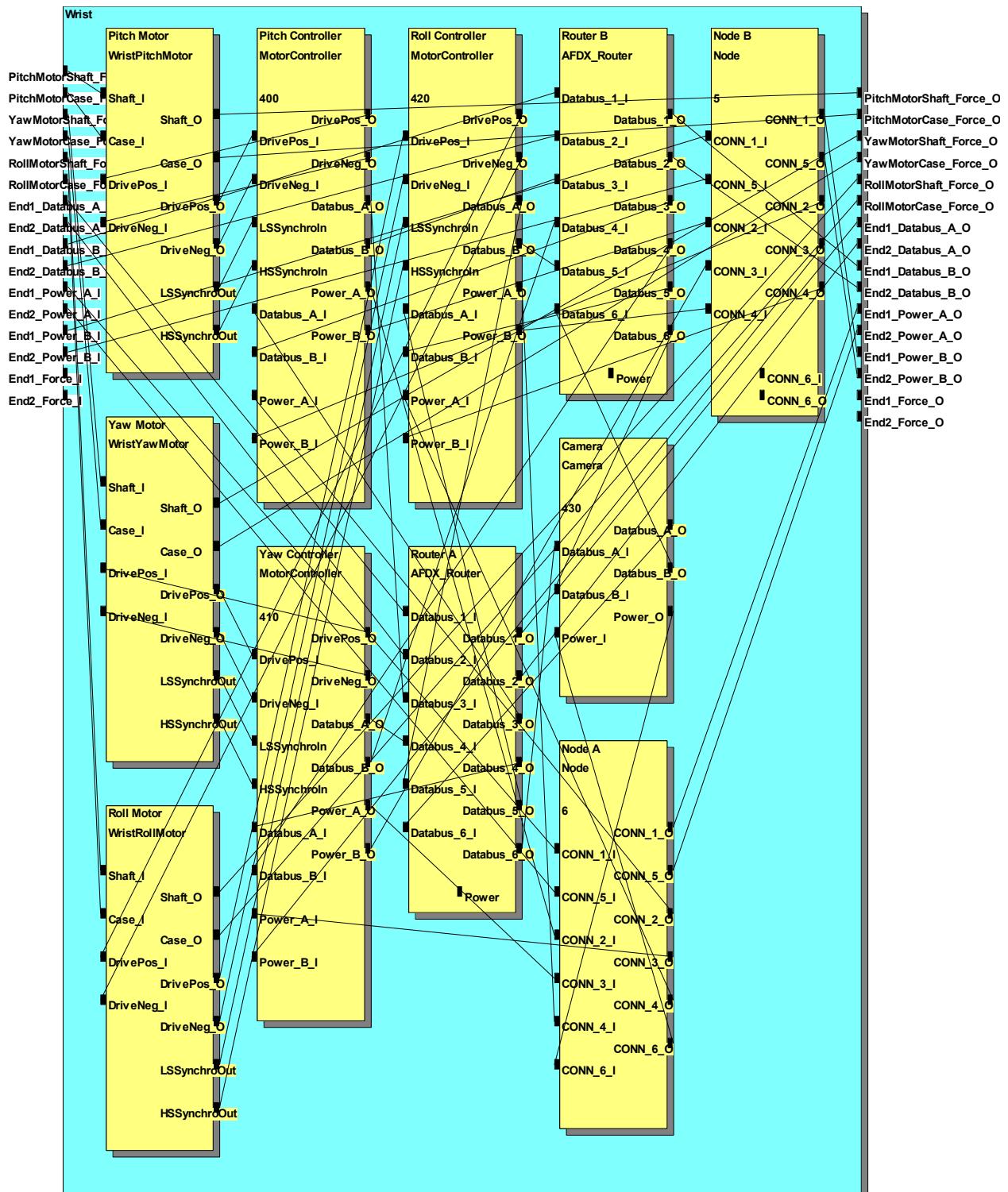
Class Name	UpperArm	
Sub-Part Class	Sub-Part Instance	
Camera	Camera	
AFDX_Router	Router_A	
	Router_B	
StrainGage	Strain Gage	
DataModule	Data Module	
Node	Node A	
	Node B	
Signal Input	Signal Output	Signal Type
End1_Databus_A_I	End1_Databus_A_O	Sig AFDX
End1_Databus_B_I	End1_Databus_B_O	Sig AFDX
End1_Force_I	End1_Force_O	Sig Force
End1_Power_A		Sig Thev
End1_Power_B		Sig Thev
End2_Databus_A_I	End2_Databus_A_O	Sig AFDX
End2_Databus_B_I	End2_Databus_B_O	Sig AFDX
End2_Force_I	End2_Force_O	Sig Force
End2_Power_A		Sig Thev
End2_Power_B		Sig Thev
Strain		Sig Double
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	Inertial Model variables	
Requirements	Send strain signals to strain gage	
Limitations	None	





3.5.4.27 Wrist

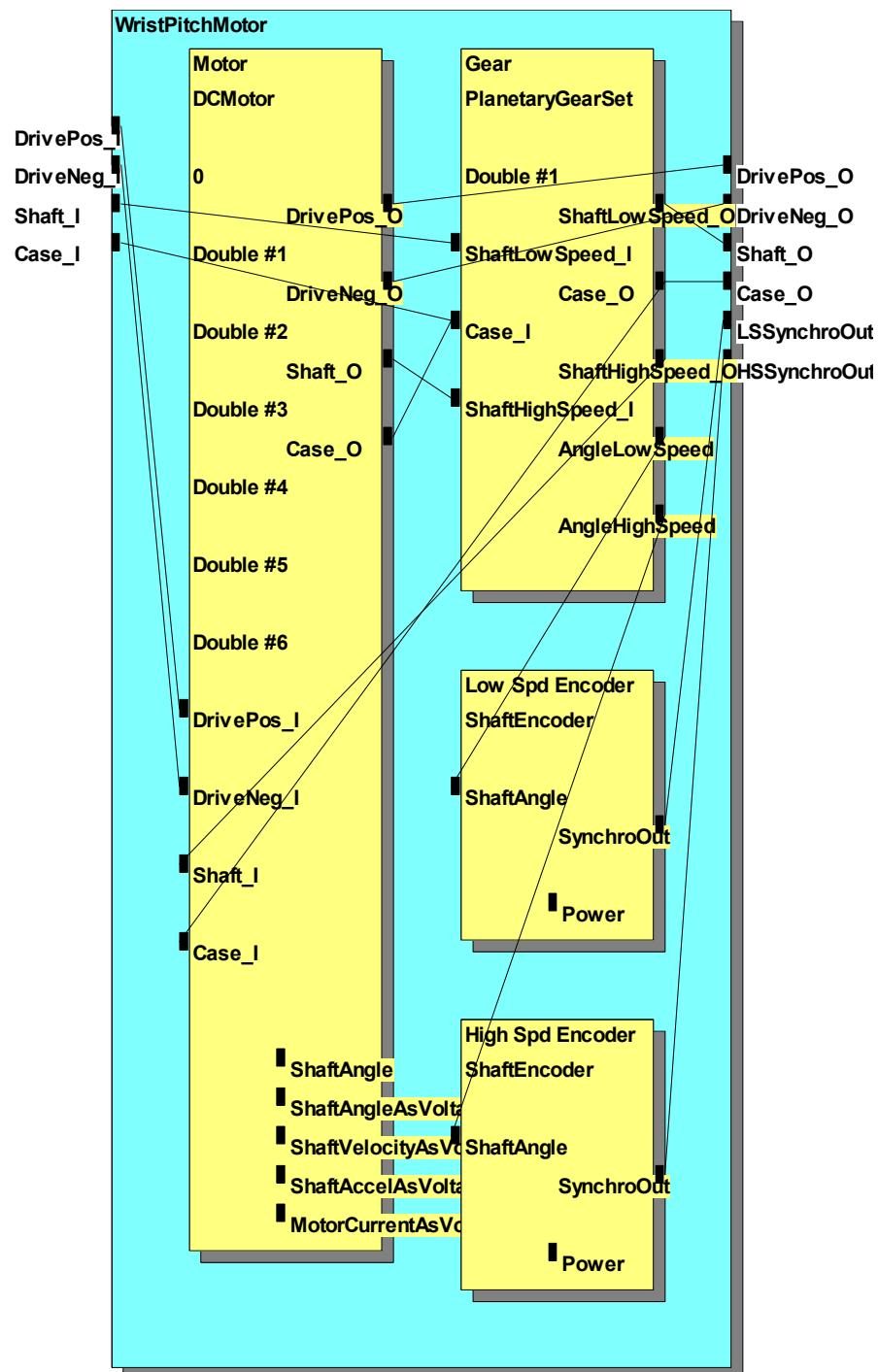
Class Name	Wrist	
Sub-Part Class	Sub-Part Instance	
ShoulderPitchMotor	Pitch Motor	
ShoulderYawMotor	Yaw Motor	
MotorController	Pitch Controller Yaw Controller	
AFDX_Router	Router A Router B	
Node	Node A Node B	
Signal Input	Signal Output	Signal Type
End1_Databus_A_I	End1_Databus_A_O	Sig AFDX
End1_Databus_B_I	End1_Databus_B_O	Sig AFDX
End1_Force_I	End1_Force_O	Sig Force
End1_Power_A		Sig Thev
End1_Power_B		Sig Thev
End2_Databus_A_I	End2_Databus_A_O	Sig AFDX
End2_Databus_B_I	End2_Databus_B_O	Sig AFDX
End2_Force_I	End2_Force_O	Sig Force
	End2_Power_A	Sig Thev
	End2_Power_B	Sig Thev
PitchMotorShaft_Force_I	PitchMotorShaft_Force_O	Sig Force
PitchMotorCase_Force_I	PitchMotorCase_Force_O	Sig Force
YawMotorShaft_Force_I	YawMotorShaft_Force_O	Sig Force
YawMotorCase_Force_I	YawMotorCase_Force_O	Sig Force
RollMotorShaft_Force_I	RollMotorShaft_Force_O	Sig Force
RollMotorCase_Force_I	RollMotorCase_Force_O	Sig Force
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.28 Wrist Pitch Motor

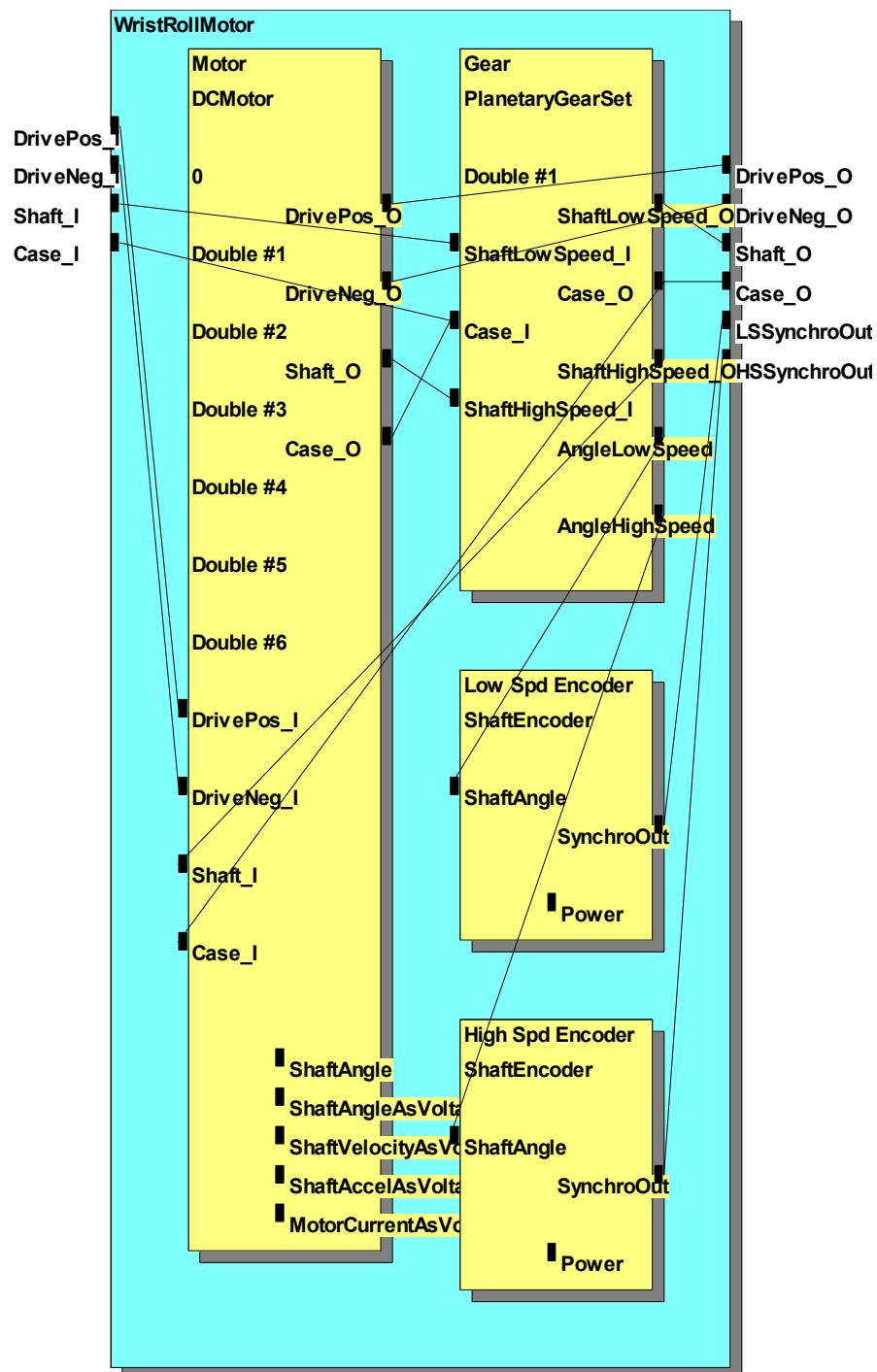
Class Name	WristPitchMotor	
Sub-Part Class	Sub-Part Instance	
DCMotor	Motor	
PlanetaryGearSet	Gear	
Shaft Encoder	Low Speed Encoder High Speed Encoder	
Signal Input	Signal Output	Signal Type
DrivePos		Sig Thev
DriveNeg		Sig Thev
Shaft		Sig Force
Case		Sig Force
	LSSynchroOut	Sig Synchro
	HSSynchroOut	Sig Synchro
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.29 Wrist Roll Motor

Class Name	WristRollMotor	
Sub-Part Class	Sub-Part Instance	
DCMotor	Motor	
PlanetaryGearSet	Gear	
Shaft Encoder	Low Speed Encoder High Speed Encoder	
Signal Input	Signal Output	Signal Type
DrivePos		Sig Thev
DriveNeg		Sig Thev
Shaft		Sig Force
Case		Sig Force
	LSSynchroOut	Sig Synchro
	HSSynchroOut	Sig Synchro
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	





3.5.4.30 Wrist Yaw Motor

Class Name	WristYawMotor	
Sub-Part Class	Sub-Part Instance	
DCMotor	Motor	
PlanetaryGearSet	Gear	
Shaft Encoder	Low Speed Encoder	
	High Speed Encoder	
Signal Input	Signal Output	Signal Type
DrivePos		Sig Thev
DriveNeg		Sig Thev
Shaft		Sig Force
Case		Sig Force
	LSSynchroOut	Sig Synchro
	HSSynchroOut	Sig Synchro
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	

